

The Hundred-Page Machine Learning Book

机器学习精讲

[加拿大] 安德烈·布可夫 (Andriy Burkov) 著

韩江雷 译



中国工信出版集团



人民邮电出版社
POSTS & TELECOM PRESS



[□□□□](#)

[□□](#)

[□□□□](#)

[□□□□](#)

[□□□](#)

[□□](#)

[□1□□□](#)

[1.1 □□□□□□](#)

[1.2 □□□□□□](#)

[1.2.1 □□□□](#)

[1.2.2 □□□□□](#)

[1.2.3 □□□□□](#)

[1.2.4 □□□□](#)

[1.3 □□□□□□□□□□](#)

[1.4 □□□□□□□□□□□□](#)

[□2□□□□□□](#)

[2.1 □□](#)

[2.1.1 □□□□](#)

[2.1.2 □□□□□\(Σ\)□□](#)

[2.1.3 □□□\(□\)□□](#)

[2.1.4 □□□□](#)

[2.1.5 □□□□](#)

[2.1.6 □□](#)

[2.1.7 max□argmax](#)

[2.1.8 □□□□□](#)

[2.1.9 時間](#)

[2.2 時間](#)

[2.3 時間](#)

[2.4 時間](#)

[2.5 時間](#)

[2.6 時間](#)

[2.7 時間vs.時間](#)

[2.8 時間時間vs.時間時間](#)

[2.9 時間vs.時間](#)

[3 時間](#)

[3.1 時間](#)

[3.1.1 時間](#)

[3.1.2 時間](#)

[3.2 時間](#)

[3.2.1 時間](#)

[3.2.2 時間](#)

[3.3 時間](#)

[3.3.1 時間](#)

[3.3.2 時間](#)

[3.4 時間](#)

[3.4.1 時間](#)

[3.4.2 時間時間](#)

[3.5 k時間](#)

[4 時間](#)

[4.1 時間時間時間](#)

[4.2 時間](#)

[4.3 時間時間時間時間](#)

[4.4 時間時間](#)

[5 時間](#)

[5.1 時間](#)

[5.1.1 時間](#)

[5.1.2 時間](#)

[5.1.3 時間](#)

[5.1.4 資料](#)

[5.1.5 評価指標](#)

[5.1.6 評価結果](#)

[5.2 評価結果](#)

[5.3 3次元](#)

[5.4 評価指標](#)

[5.5 資料](#)

[5.6 評価指標](#)

[5.6.1 評価指標](#)

[5.6.2 評価指標/評価指標](#)

[5.6.3 評価指標](#)

[5.6.4 評価指標](#)

[5.6.5 ROC評価指標](#)

[5.7 評価指標](#)

[評価指標](#)

[6 評価指標](#)

[6.1 評価指標](#)

[6.1.1 評価指標](#)

[6.1.2 評価指標](#)

[6.2 評価指標](#)

[6.2.1 評価指標](#)

[6.2.2 評価指標](#)

[7 評価指標](#)

[7.1 評価指標](#)

[7.2 評価指標](#)

[7.3 評価指標](#)

[7.4 評価指標](#)

[7.5 評価指標](#)

[7.5.1 評価指標](#)

[7.5.2 評価指標](#)

[7.5.3 評価指標](#)

[7.6 評価指標](#)

[7.7 評価指標](#)

[7.8 評価指標](#)

[7.9 評価](#)

[7.10 評価](#)

[7.11 評価](#)

[8 評価](#)

[8.1 評価](#)

[8.2 評価](#)

[8.3 評価](#)

[8.4 評価](#)

[8.5 評価](#)

[8.6 評価](#)

[8.7 評価](#)

[8.8 評価](#)

[9 評価](#)

[9.1 評価](#)

[9.2 評価](#)

[9.2.1 k](#)

[9.2.2 DBSCAN HDBSCAN](#)

[9.2.3 評価](#)

[9.2.4 評価](#)

[9.3 評価](#)

[9.3.1 評価](#)

[9.3.2 UMAP](#)

[9.4 評価](#)

[10 評価](#)

[10.1 評価](#)

[10.2 評価](#)

[10.3 評価](#)

[10.3.1 評価](#)

[10.3.2 評価](#)

[10.4 評価: 評価](#)

[11 評価](#)

[11.1 評価](#)

11.2 □□□□

11.3 □□□□□□

11.4 □□□□□

11.5 □□□□□□□□□□

11.6 □□□□

11.7 □□□□

□□□

□□□□

□□□□□□□□

ISBN□978-7-115-51853-8

□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□



□ [□□□] □□□·□□□□Andriy Burkov□

□ □□□

□□□□ □□□

□□□□□□□□□□ □□□□□□□□□11□

□□ 100164 □□□□ 315@ptpress.com.cn

□□ <http://www.ptpress.com.cn>

□□□□□□(010)81055410

□□□□□□(010)81055315



Simplified Chinese translation copyright ©2019 by Posts and Telecommunications Press.

ALL RIGHTS RESERVED.

The Hundred-Page Machine Learning Book by Andriy Burkov. Copyright © 2019 Andriy Burkov.

Andriy Burkov
Machine Learning Book

Machine Learning Book



「認知」の定義は、人間の知能を模倣し、人間の知能が実行するタスクを実行する能力を指す。これは、人間の知能を模倣する能力を指す。これは、人間の知能を模倣する能力を指す。

認知科学は、人間の知能の仕組みを理解するための学問である。認知科学は、人間の知能の仕組みを理解するための学問である。認知科学は、人間の知能の仕組みを理解するための学問である。

「認知」の定義は、人間の知能を模倣し、人間の知能が実行するタスクを実行する能力を指す。これは、人間の知能を模倣する能力を指す。これは、人間の知能を模倣する能力を指す。これは、人間の知能を模倣する能力を指す。これは、人間の知能を模倣する能力を指す。これは、人間の知能を模倣する能力を指す。

認知科学は、人間の知能の仕組みを理解するための学問である。認知科学は、人間の知能の仕組みを理解するための学問である。認知科学は、人間の知能の仕組みを理解するための学問である。

認知科学

2060



第1章 绪论

1.1 机器学习概述

机器学习是人工智能的一个分支，它研究如何使计算机在没有明确编程的情况下，通过经验数据自动学习并改进其性能。机器学习的应用非常广泛，包括自然语言处理、计算机视觉、推荐系统等。

机器学习可以分为监督学习、无监督学习和强化学习三大类。监督学习是指通过已知的输入输出对来训练模型，使其能够对新的输入数据进行预测。无监督学习是指通过寻找数据中的内在结构来训练模型，如聚类分析。强化学习是指通过与环境交互来学习最优策略，以最大化累积奖励。

在机器学习中，我们通常将数据分为训练集和测试集。训练集用于训练模型，而测试集用于评估模型的性能。评估指标通常包括准确率、召回率、F1分数等。

1.2 机器学习的应用

机器学习在多个领域都有广泛的应用。在工业领域，机器学习可以用于设备故障预测、质量控制等。在金融领域，机器学习可以用于信用风险评估、欺诈检测等。在医疗领域，机器学习可以用于疾病诊断、药物研发等。在农业领域，机器学习可以用于作物产量预测、病虫害检测等。

1.2.1 监督学习

监督学习(supervised learning)^[1]是指通过已知的输入输出对(dataset)，训练模型使其能够对新的输入数据进行预测。监督学习通常分为分类任务和回归任务。在分类任务中，模型需要预测输入数据的类别。在回归任务中，模型需要预测输入数据的连续值。监督学习的核心是找到一个函数，使得该函数能够最好地拟合训练数据，并在新数据上具有良好的泛化能力。

半监督学习(semi-supervised learning)介于监督学习和无监督学习之间,它利用少量标注数据和大量未标注数据,通过半监督学习算法(semi-supervised learning algorithm)来学习模型,从而提高模型的泛化能力。

例如,在文本分类任务中,我们可能只有少量标注的文本,但拥有大量未标注的文本。通过半监督学习,我们可以利用未标注的文本来学习模型,从而提高模型的分类性能。

1.2.4 强化学习

强化学习(reinforcement learning)是一种通过试错来学习模型的方法。它通常涉及一个智能体(agent)和一个环境(environment)。智能体通过观察环境的状态(state)来做出决策,并根据决策的结果获得奖励或惩罚。通过不断的学习,智能体可以学会在环境中做出最优的决策。



例如,在游戏任务中,智能体通过观察游戏状态来做出决策,并根据得分获得奖励。通过不断的学习,智能体可以学会在游戏中做出最优的决策。

但是，在训练模型之前，我们需要对数据进行预处理。预处理包括分词、去除停用词、词干提取等。在本节中，我们将介绍如何对文本数据进行预处理。

1.3 文本数据预处理

在训练模型之前，我们需要对数据进行预处理。预处理包括分词、去除停用词、词干提取等。在本节中，我们将介绍如何对文本数据进行预处理。

分词是将文本拆分成单词的过程。在Python中，我们可以使用正则表达式来实现分词。例如，我们可以使用以下代码来分词：

```
import re
tokens = re.findall(r'\w+', text.lower())
```

其中，`tokens`是一个列表，包含了文本中的所有单词。我们使用`lower()`方法将文本转换为小写字母，以便统一处理。

去除停用词是指去除那些对模型没有贡献的单词。例如，在自然语言处理中，我们经常遇到一些常见的单词，如“the”、“is”、“and”等，这些单词被称为停用词。我们可以使用以下代码来去除停用词：

在训练模型之前，我们需要对数据进行预处理。预处理包括分词、去除停用词、词干提取等。在本节中，我们将介绍如何对文本数据进行预处理。

- 将文本转换为小写字母，以便统一处理。
- 去除停用词。
- 词干提取。

在训练模型之前，我们需要对数据进行预处理。预处理包括分词、去除停用词、词干提取等。在本节中，我们将介绍如何对文本数据进行预处理。

例如, 我们考虑一个二分类问题, 即判断一封邮件是否是垃圾邮件。我们可以将垃圾邮件记作 0, 非垃圾邮件记作 1。我们使用“支持向量机”(Support Vector Machine, SVM) 来求解这个问题。SVM 的目标是找到一个超平面 (hyperplane), 使得两类数据点能够被正确地分开。假设我们有一个特征向量 x , 我们希望找到一个权重向量 w 和一个偏置 b , 使得对于垃圾邮件 (0), 有 $w \cdot x - b \leq -1$; 对于非垃圾邮件 (1), 有 $w \cdot x - b \geq 1$ 。

在 SVM 中, 我们通常使用拉格朗日乘子法来求解优化问题, 从而找到最优的超平面。

SVM 的一个重要特性是它能够处理非线性可分的数据。通过引入核函数 (kernel function), SVM 可以将数据映射到高维空间, 从而找到最优的超平面。例如, 如果我们使用高斯核函数 (Gaussian kernel), 那么 SVM 可以处理非线性可分的数据。假设我们有一个数据集, 其中垃圾邮件和非垃圾邮件的分布是非线性的。通过引入核函数, SVM 可以将数据映射到高维空间, 从而找到最优的超平面。

在 SVM 中, 我们通常使用拉格朗日乘子法来求解优化问题, 从而找到最优的超平面。假设我们有一个特征向量 x , 我们希望找到一个权重向量 w 和一个偏置 b , 使得对于垃圾邮件 (0), 有 $w \cdot x - b \leq -1$; 对于非垃圾邮件 (1), 有 $w \cdot x - b \geq 1$ 。

$$w \cdot x - b = 0$$

其中, $w \cdot x$ 表示权重向量 w 与特征向量 x 的内积。假设我们有一个数据集, 其中垃圾邮件和非垃圾邮件的分布是非线性的。通过引入核函数, SVM 可以将数据映射到高维空间, 从而找到最优的超平面。

在 SVM 中, 我们通常使用拉格朗日乘子法来求解优化问题, 从而找到最优的超平面。假设我们有一个特征向量 x , 我们希望找到一个权重向量 w 和一个偏置 b , 使得对于垃圾邮件 (0), 有 $w \cdot x - b \leq -1$; 对于非垃圾邮件 (1), 有 $w \cdot x - b \geq 1$ 。

在 SVM 中, 我们通常使用拉格朗日乘子法来求解优化问题, 从而找到最优的超平面。假设我们有一个特征向量 x , 我们希望找到一个权重向量 w 和一个偏置 b , 使得对于垃圾邮件 (0), 有 $w \cdot x - b \leq -1$; 对于非垃圾邮件 (1), 有 $w \cdot x - b \geq 1$ 。

$$y = \text{sign}(w \cdot x - b)$$

其中, sign 表示符号函数, 用于判断一个数值的正负。假设我们有一个数据集, 其中垃圾邮件和非垃圾邮件的分布是非线性的。通过引入核函数, SVM 可以将数据映射到高维空间, 从而找到最优的超平面。

SVM 模型参数为 \mathbf{w} 和 b ，最优参数为 \mathbf{w}^* 和 b^* ，模型函数为 $f(\mathbf{x})$ ，表示为：

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^* \mathbf{x} - b^*)$$

其中， sign 函数表示符号函数，当输入为正时输出为 +1，当输入为负时输出为 -1。即 $\text{sign}(x) = +1$ 当 $x \geq 0$ ， $\text{sign}(x) = -1$ 当 $x < 0$ 。

那么，如何找到最优参数 \mathbf{w}^* 和 b^* ？这通常通过优化（optimization）方法来实现。

例如，对于线性可分数据，可以通过求解以下优化问题来找到最优参数：

$$\min_{\mathbf{w}, b} \sum_{i=1, \dots, 10000} \max(0, 1 - y_i(\mathbf{w} \mathbf{x}_i - b))$$

其中 \mathbf{x}_i 和 y_i 分别是第 i 个样本的特征向量和类别标签（+1 或 -1）。

$$\begin{aligned} \mathbf{w} \mathbf{x}_i - b &\geq +1, & \text{当 } y_i = +1 \text{ 时} \\ \mathbf{w} \mathbf{x}_i - b &\leq -1, & \text{当 } y_i = -1 \text{ 时} \end{aligned}$$

此外，SVM 还关注模型的泛化能力（generalization），即模型在未见数据上的表现。

在 SVM 中，权重向量 \mathbf{w} 的欧几里得范数（Euclidean norm） $\|\mathbf{w}\|$ 是一个重要的参数，其定义为：

$$\|\mathbf{w}\| = \sqrt{\sum_{j=1}^D (w^{(j)})^2}$$

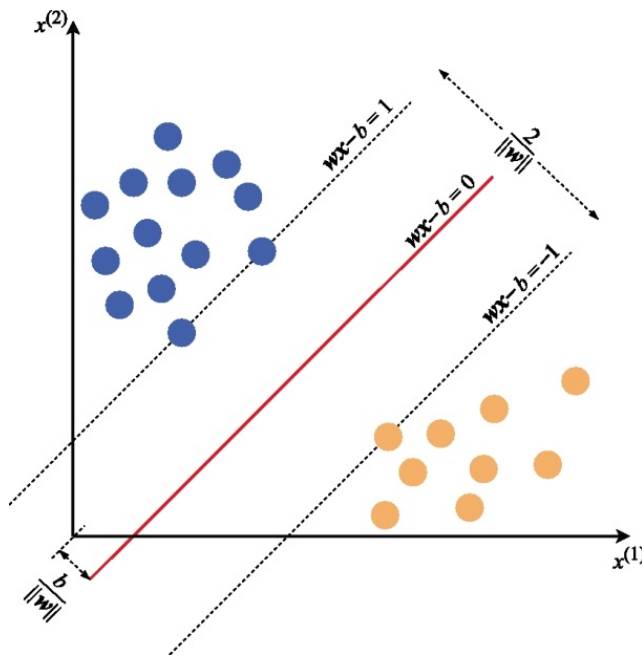
因此，SVM 的优化问题可以表述为：

$$\begin{aligned} &\text{minimize } \|\mathbf{w}\| \\ &\text{s. t. } y_i(\mathbf{w} \mathbf{x}_i - b) \geq 1, \quad \text{对于 } i = 1, \dots, N \end{aligned}$$

即, 即 $y_i(\mathbf{w}\mathbf{x}_i - b) \geq 1$ 即即即即即即即

即即即即 \mathbf{w}^* 即 b^* 即即即即 (statistical model), 即即即即即即即即即
即即即即 (training) 即

即 1.1 即即即 SVM 即即即即即 1.1 即, 即即即即即即即, 即即即即即即即即
即即即; 即即即即即, 即 $\mathbf{w}\mathbf{x} - b = 0$ 即即



即 1.1 即即即即即即即即即 SVM 即即即

即, 即即 \mathbf{w} 即即即即即即即即即即即即即即即即即? 即 1.1 即, $\mathbf{w}\mathbf{x} - b = 1$ 即 $\mathbf{w}\mathbf{x} - b = -1$ 即即即即即即即即即即即即即, 即即即即即即 $\frac{\|\mathbf{w}\|_2}{2}$ 即即, 即 $\|\mathbf{w}\|_2$ 即, 即即即即即即即

即即即即即即即即即即即即即即即即即即即 (linear model) 即即即
即即, 即即即即即即即即即 (即即即即即即即即即) 即 SVM 即即即即即即即

(kernel) 可以將輸入空間中的資料點映射到更高維度的空間，使得原本在低維度空間中無法線性分離的資料，在高維度空間中可以線性分離。而 (outlier) (異常值) 是指那些與大多數資料點分離的點，這些點在 SVM 模型中通常被視為誤分類。[5] 在 SVM 模型中，通常使用 3 個參數來描述 SVM 模型。

在 SVM 模型中，通常使用 3 個參數來描述 SVM 模型：核函數 (kernel)、核係數 (gamma) 和軟邊界參數 (C)。核函數 (kernel) 是指將輸入空間中的資料點映射到更高維度空間的函數。核係數 (gamma) 是指核函數中的參數。軟邊界參數 (C) 是指 SVM 模型中的參數，它控制了模型的複雜度。

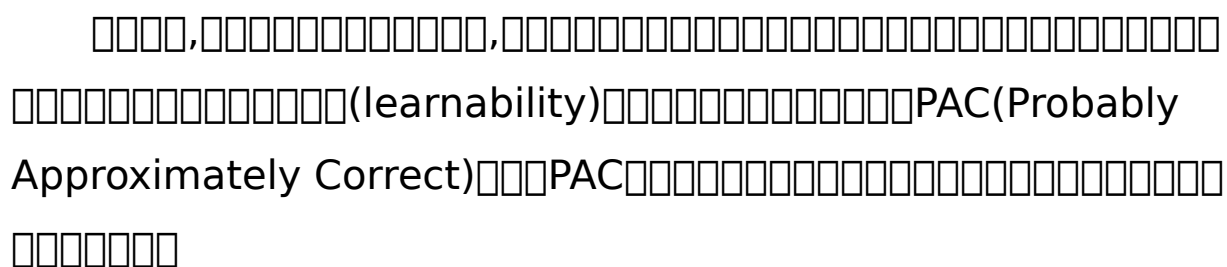
在 SVM 模型中，通常使用 3 個參數來描述 SVM 模型：核函數 (kernel)、核係數 (gamma) 和軟邊界參數 (C)。核函數 (kernel) 是指將輸入空間中的資料點映射到更高維度空間的函數。核係數 (gamma) 是指核函數中的參數。軟邊界參數 (C) 是指 SVM 模型中的參數，它控制了模型的複雜度。

1.4 核函數與核係數

在 SVM 模型中，核函數 (kernel) 是指將輸入空間中的資料點映射到更高維度空間的函數。核係數 (gamma) 是指核函數中的參數。軟邊界參數 (C) 是指 SVM 模型中的參數，它控制了模型的複雜度。在 SVM 模型中，核函數 (kernel) 是指將輸入空間中的資料點映射到更高維度空間的函數。核係數 (gamma) 是指核函數中的參數。軟邊界參數 (C) 是指 SVM 模型中的參數，它控制了模型的複雜度。

在 SVM 模型中，核函數 (kernel) 是指將輸入空間中的資料點映射到更高維度空間的函數。核係數 (gamma) 是指核函數中的參數。軟邊界參數 (C) 是指 SVM 模型中的參數，它控制了模型的複雜度。

在 SVM 模型中，核函數 (kernel) 是指將輸入空間中的資料點映射到更高維度空間的函數。核係數 (gamma) 是指核函數中的參數。軟邊界參數 (C) 是指 SVM 模型中的參數，它控制了模型的複雜度。



- [illegible]

2 〇〇〇〇〇

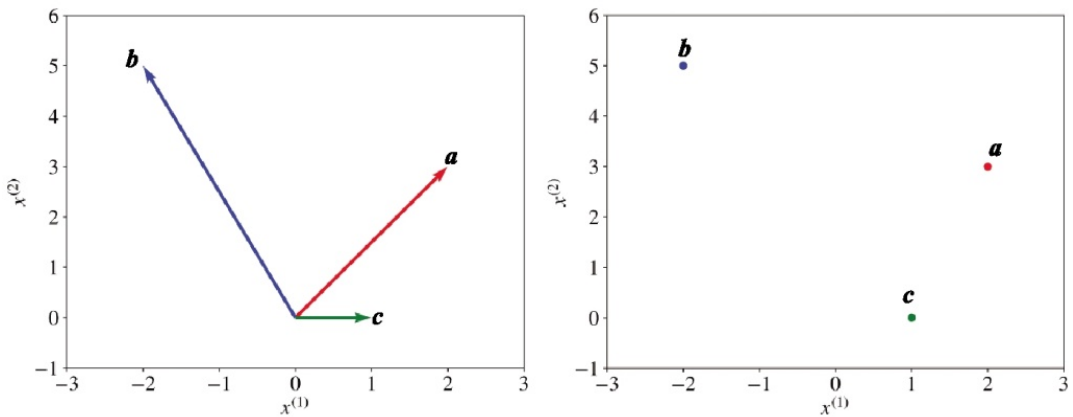
2.1 〇〇

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

2.1.1 〇〇〇〇

□□□(scalar)□□□□□□,□15□□-3.25□□□□□□□□□□□□□□□□□□
 □□□,□□ $x \square a$

`(vector)`和`(ordered list)`,
`x`和`w`,
 2.1和3,
`a=[2,3]`和`b=[-2,5]`和`c=[1,0]`,
 $w^{(j)}$ 和 $x^{(j)}$,
 \boldsymbol{a} , $\boldsymbol{a}^{(1)}$ 和 $\boldsymbol{a}^{(2)}$



□2.1 3□□□□□□□□3□□□□□3□□

$$x^{(j)}, x^2(\cdot), x^3(\cdot) \text{ 和 } x^{(j)2}$$
$$\mathbb{R}^n, x_i^{(j)}, x_{i,j}^{(k)}, x_{l,u}^{(j)}, \mu, \nu$$

matrix(array)

$$\begin{bmatrix} 2 & 4 & -3 \\ 21 & -6 & -1 \end{bmatrix}$$

□□□□□□□□□□□□, □□ **A W** □

unordered(set) unordered unordered unordered(unordered collection) unordered
 unordered(S) unordered unordered unordered unordered unordered unordered unordered, unordered
 $\{1,3,18,23,235\}$ unordered $\{x_1,x_2,x_3,x_4,\dots,x_n\}$ unordered, unordered unordered unordered
 unordered unordered unordered a unordered b unordered, unordered a unordered b unordered, unordered unordered unordered unordered $[a,b]$;
 unordered a unordered b unordered, unordered unordered unordered (a,b) unordered, unordered $[0,1]$ unordered unordered:0
 0.000 1 0.25 0.784 0.999 5 1.0 unordered, unordered \mathbb{R} unordered unordered unordered, unordered
 unordered unordered unordered unordered unordered

$x \in S$ “ x belongs to S ” $S_1 \cap S_2$ (intersection) $S_1 \cup S_2$ (union)
 $S_3, S_3 \leftarrow S_1 \cap S_2, \{1, 8\} \leftarrow \{1, 3, 5, 8\} \cap \{1, 8, 4\}$

$S_1 \cup S_2$ (union) $S_3, S_3 \leftarrow S_1 \cup S_2$,
 $\{1,3,4,5,8\} \leftarrow \{1,3,5,8\} \cup \{1,8,4\}$

2.1.2 Summation (Σ)

Let $\mathbf{X} = \{x_1, x_2, \dots, x_{n-1}, x_n\}$, then $\mathbf{x} = [x^{(1)}, x^{(2)}, \dots, x^{(m-1)}, x^{(m)}]$ and Σ :

$$\sum_{i=1}^n x_i \stackrel{\text{def}}{=} x_1 + x_2 + \dots + x_{n-1} + x_n$$

or

$$\sum_{j=1}^m x^{(j)} \stackrel{\text{def}}{=} x^{(1)} + x^{(2)} + \dots + x^{(m-1)} + x^{(m)}$$

or, $\stackrel{\text{def}}{=}$ means “is defined as”

2.1.3 Product (Π)

Π is the product of Σ and Π is the product of Σ :

$$\prod_{i=1}^n x_i \stackrel{\text{def}}{=} x_1 \cdot x_2 \cdot \dots \cdot x_{n-1} \cdot x_n$$

or, $a \cdot b$ is a times b , or ab is a times b

2.1.4

集合的平方根: $S' \leftarrow \{x^2 \mid x \in S, x > 3\}$ 集合的平方根, 集合 S 中所有大于 3 的元素的平方, 集合 S' 的大小 $|S'|$ 等于集合 S 中所有大于 3 的元素的平方

2.1.5 向量

向量加法: $\mathbf{x} + \mathbf{z} = [x^{(1)} + z^{(1)}, x^{(2)} + z^{(2)}, \dots, x^{(m)} + z^{(m)}]$

向量减法: $\mathbf{x} - \mathbf{z} = [x^{(1)} - z^{(1)}, x^{(2)} - z^{(2)}, \dots, x^{(m)} - z^{(m)}]$

向量的标量乘法:

$$\mathbf{x}c \stackrel{\text{def}}{=} [cx^{(1)}, cx^{(2)}, \dots, cx^{(m)}]$$

点积(dot product)或内积(inner product)运算, 即

$$\mathbf{w} \cdot \mathbf{x} \stackrel{\text{def}}{=} \sum_{i=1}^m w^{(i)} x^{(i)}$$

向量的点积 $\mathbf{w} \cdot \mathbf{x}$ 等于向量的对应元素的乘积之和, 即 $w^{(1)}x^{(1)} + w^{(2)}x^{(2)} + \dots + w^{(m)}x^{(m)}$

向量的点积

向量的点积 $\mathbf{w} \cdot \mathbf{x}$ 等于向量的对应元素的乘积之和, 即 $w^{(1)}x^{(1)} + w^{(2)}x^{(2)} + \dots + w^{(m)}x^{(m)}$

$$\mathbf{W} = \begin{bmatrix} w^{(1,1)} & w^{(1,2)} & w^{(1,3)} \\ w^{(2,1)} & w^{(2,2)} & w^{(2,3)} \end{bmatrix}$$

向量的点积 $\mathbf{w} \cdot \mathbf{x}$ 等于向量的对应元素的乘积之和, 即 $w^{(1)}x^{(1)} + w^{(2)}x^{(2)} + \dots + w^{(m)}x^{(m)}$

向量的点积 $\mathbf{w} \cdot \mathbf{x}$ 等于向量的对应元素的乘积之和, 即 $w^{(1)}x^{(1)} + w^{(2)}x^{(2)} + \dots + w^{(m)}x^{(m)}$

向量的点积 $\mathbf{w} \cdot \mathbf{x}$ 等于向量的对应元素的乘积之和, 即 $w^{(1)}x^{(1)} + w^{(2)}x^{(2)} + \dots + w^{(m)}x^{(m)}$

, **w**,

□□□□□,□□□□□□□□□□,□□□□(transpose)□□□□□□□□□□

$$\mathbf{x}^T \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{x} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}:$$

$$x = \begin{bmatrix} x^{(1)} \\ x^{(2)} \end{bmatrix}, \quad x^T \stackrel{\text{def}}{=} [x^{(1)}, x^{(2)}]$$

$$\mathbf{x}^T \mathbf{W} \mathbf{x}^T \mathbf{W} :$$

$$\mathbf{x}^T \mathbf{W} = [x^{(1)}, x^{(2)}] \begin{bmatrix} w^{(1,1)} & w^{(1,2)} & w^{(1,3)} \\ w^{(2,1)} & w^{(2,2)} & w^{(2,3)} \end{bmatrix}$$

$$\stackrel{\text{def}}{=} [w^{(1,1)}x^{(1)} + w^{(2,1)}x^{(2)}, w^{(1,2)}x^{(1)} + w^{(2,2)}x^{(2)}, w^{(1,3)}x^{(1)} + w^{(2,3)}x^{(2)}]$$

□□□□□□, □□□□□□□□□□□□□□□□, □□□□□□□□□□

2.1.6

1. f is a function from X to Y if and only if X is a non-empty set (domain), Y is a non-empty set (co-domain), and for every $x \in X$, there exists a unique $y \in Y$ such that $y = f(x)$. We write $f: X \rightarrow Y$.

自变量(variable) x 函数 f

如果 $x=c$ 满足 $f(x) \geq f(c)$, 那么 $x=c$ 是 $f(x)$ 的局部极小值 (local minimum) 在区间 (interval) 上: 开区间 (open interval) 闭区间: $(0,1)$ 开区间 0 到 1 闭区间 0 到 1 , 图 2.2

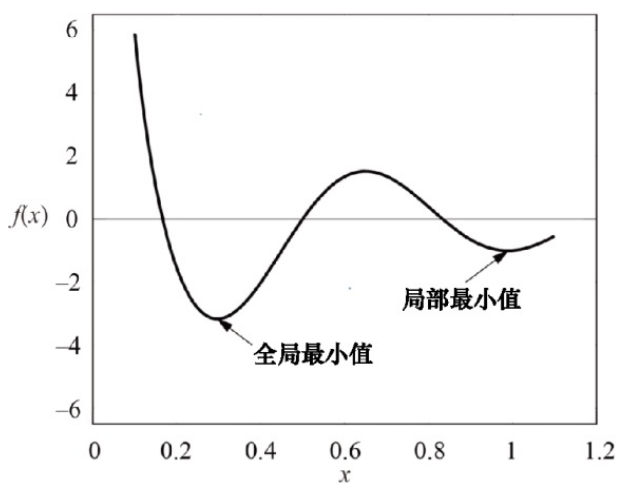


图 2.2 极值点

$y=f(x)$ 的极值点 y 的极值点

2.1.7 max 和 argmax

设 $A=\{a_1, a_2, \dots, a_n\}$, 那么 $\max_{a \in A} f(a)$ 表示: A 中所有 $f(a)$ 的最大值 $\arg\max_{a \in A} f(a)$ 表示 $f(a)$ 取得最大值 a

如果 f 是实数函数, 那么 $\max_a f(a)$ 和 $\arg\max_a f(a)$

$$\min \argmin \max \argmax$$

2.1.8 〇〇〇〇〇

$$a \leftarrow f(x), \quad a \leftarrow f(x) \quad a$$
[illegible]

2.1.9 五五五五五

1. 微分係数 (derivative) $f'(x)$ (導関数) は、関数 $f(x)$ の傾きの関数である。

 2. 例: $f(x) = x^2$ の場合、 $f'(x) = 2x$ である。

 3. 微分係数は、関数の傾きを表す。

 4. 微分係数は、関数の傾きの関数である。

 5. 微分係数は、関数の傾きの関数である。

 6. 微分係数は、関数の傾きの関数である。

 7. 微分係数は、関数の傾きの関数である。

 8. 微分係数は、関数の傾きの関数である。

 9. 微分係数は、関数の傾きの関数である。

 10. 微分係数は、関数の傾きの関数である。

□□□□□□□□□□(differentiation)□

$f(x) = x^2, f'(x) = 2x; f(x) = 2x, f'(x) = 2; f(x) = 2, f'(x) = 0$ (konstante Funktion)

□□□□□□□□(chain rule)□□□□□□□□□□□□, □ $F(x) = f(g(x))$, f □
 g □□□□, □ $F'(x) = f'(g(x))g'(x)$ □□□□□□, □□ $F(x) = (5x+1)^2$, □□
 $g(x) = 5x+1$ □ $f(g(x)) = (g(x))^2$ □□□□□□□□, □□□□
 $F'(x) = 2(5x+1)g'(x) = 2 \times (5x+1) \times 5 = 50x+10$ □

$\mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x}|\mathbf{z})} \mathbb{E}_{\mathbf{y} \sim p(\mathbf{y}|\mathbf{x})} \mathbb{E}_{\mathbf{z}' \sim p(\mathbf{z})} \mathbb{E}_{\mathbf{x}' \sim p(\mathbf{x}|\mathbf{z})} \mathbb{E}_{\mathbf{y}' \sim p(\mathbf{y}|\mathbf{x})} \mathbb{E}_{\mathbf{z}'' \sim p(\mathbf{z})} \mathbb{E}_{\mathbf{x}'' \sim p(\mathbf{x}|\mathbf{z})} \mathbb{E}_{\mathbf{y}'' \sim p(\mathbf{y}|\mathbf{x})} \mathbb{E}_{\mathbf{z}'''} \mathbb{E}_{\mathbf{x}'''} \mathbb{E}_{\mathbf{y}'''}$
 (generalization) $\mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x}|\mathbf{z})} \mathbb{E}_{\mathbf{y} \sim p(\mathbf{y}|\mathbf{x})} \mathbb{E}_{\mathbf{z}' \sim p(\mathbf{z})} \mathbb{E}_{\mathbf{x}' \sim p(\mathbf{x}|\mathbf{z})} \mathbb{E}_{\mathbf{y}' \sim p(\mathbf{y}|\mathbf{x})} \mathbb{E}_{\mathbf{z}'' \sim p(\mathbf{z})} \mathbb{E}_{\mathbf{x}'' \sim p(\mathbf{x}|\mathbf{z})} \mathbb{E}_{\mathbf{y}'' \sim p(\mathbf{y}|\mathbf{x})} \mathbb{E}_{\mathbf{z}'''} \mathbb{E}_{\mathbf{x}'''} \mathbb{E}_{\mathbf{y}'''}$

[illegible]

概率分布(probability distribution), 离散型
 概率分布, 离散型概率函数(probability mass function, pmf) 为
 $\Pr(X=x_1)=0.3, \Pr(X=x_2)=0.45, \Pr(X=x_3)=0.25$ 的概率分布
 函数, 其值域为 $[0, 1]$ [图 2.3(a)]

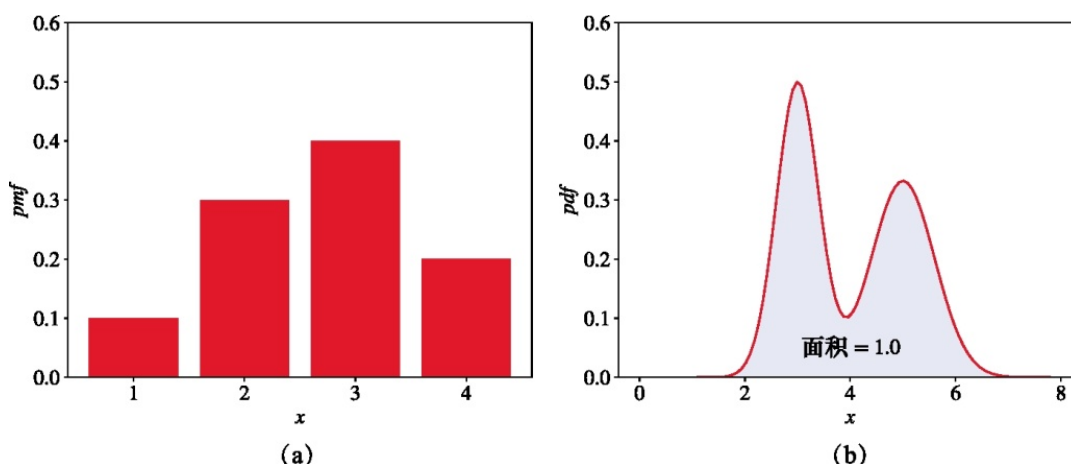


图 2.3 离散型概率分布

连续型随机变量(continuous random variable)的概率分布函数
 为 $F(x)$, 其值域为 $[0, 1]$. 连续型随机变量 X 的概率密度函数, 记为 $f(x)$
 $\Pr(X=c)=0$ 的概率密度函数(probability density function, pdf) 为
 非负函数, 其值域为 $[0, 1]$ [图 2.3(b)]

连续型随机变量 X 的期望, 记为 $\mathbb{E}[X]$ (expectation):

$$\mathbb{E}[X] \stackrel{\text{def}}{=} \sum_{i=1}^k x_i \cdot \Pr(X = x_i) = x_1 \cdot \Pr(X = x_1) + x_2 \cdot \Pr(X = x_2) + \dots + x_k \cdot \Pr(X = x_k)$$

(2.1)

$\Pr(X=\mathbf{x}_i)$ pmf, X x , (mean, average), μ (statistics)

(standard deviation),

$$\sigma \stackrel{\text{def}}{=} \sqrt{\mathbb{E} [(X - \mu)^2]}$$

(variance), $\sigma^2 = \text{var}(X)$,

$$\sigma^2 = \mathbb{E} [(X - \mu)^2]$$

:

$$\sigma = \sqrt{\Pr(X = x_1) (x_1 - \mu)^2 + \Pr(X = x_2) (x_2 - \mu)^2 + \dots + \Pr(X = x_k) (x_k - \mu)^2} \tag{2.2}$$

$$\mu = \mathbb{E}[X]$$

X :

$$\mathbb{E} [X] \stackrel{\text{def}}{=} \int_{\mathbb{R}} x f_X(x) \, dx \tag{2.3}$$

f_X pdf, $\int_{\mathbb{R}} x f_X(x) \, dx$ (integral)

, $\int_{\mathbb{R}} f_X(x) \, dx = 1$

f_X (example) (sample) (dataset)

2.3

$S_X = \{x_i\}_{i=1}^N$ (unbiased estimator)

S_X , θ , $\hat{\theta}(S_X)$

$$\mathbb{E}[\hat{\theta}(S_X)] = \theta$$

$\hat{\theta}$ (sample statistic), $\hat{\theta}$ S_X , θ X , $\hat{\mu}$, μ

2.1 2.3 $\mathbb{E}[X]$ $\frac{1}{N} \sum_{i=1}^N x_i$ (sample mean)

2.4

(conditional probability) $Y=y$, $X=x$, $\Pr(X=x|Y=y)$ (Bayes'Rule) (Bayes'Theorem),

$$\Pr(X = x|Y = y) = \frac{\Pr(Y = y|X = x)\Pr(X = x)}{\Pr(Y = y)}$$

2.5 高斯分布

高斯分布是高维数据中最重要的分布之一，其概率密度函数 f_{θ} 由均值 μ 和方差 σ 决定 (Gaussian function)。

$$f_{\theta}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

其中， $\theta \stackrel{\text{def}}{=} [\mu, \sigma]$ 。

高斯分布的一个重要性质是，其参数 θ 可以通过观测数据 X 来估计。具体来说，给定观测数据 X ，参数 θ 的估计值 $\hat{\theta}$ 为：

$$\begin{aligned} \Pr(\theta = \hat{\theta}|X = x) &\leftarrow \frac{\Pr(X = x|\theta = \hat{\theta}) \Pr(\theta = \hat{\theta})}{\Pr(X = x)} \\ &= \frac{\Pr(X = x|\theta = \hat{\theta}) \Pr(\theta = \hat{\theta})}{\sum_{\tilde{\theta}} \Pr(X = x|\theta = \tilde{\theta}) \Pr(\theta = \tilde{\theta})} \end{aligned}$$

其中， $\Pr(X = x|\theta = \hat{\theta}) \stackrel{\text{def}}{=} f_{\hat{\theta}}$ 。

高斯分布的一个重要性质是，其参数 θ 可以通过观测数据 X 来估计。具体来说，给定观测数据 X ，参数 θ 的估计值 $\hat{\theta}$ 为：

高斯分布的一个重要性质是，其参数 θ 可以通过观测数据 X 来估计。具体来说，给定观测数据 X ，参数 θ 的估计值 $\hat{\theta}$ 为：

高斯分布的一个重要性质是，其参数 θ 可以通过观测数据 X 来估计。具体来说，给定观测数据 X ，参数 θ 的估计值 $\hat{\theta}$ 为：

高斯分布的一个重要性质是，其参数 θ 可以通过观测数据 X 来估计。具体来说，给定观测数据 X ，参数 θ 的估计值 $\hat{\theta}$ 为：

高斯分布的一个重要性质是，其参数 θ 可以通过观测数据 X 来估计。具体来说，给定观测数据 X ，参数 θ 的估计值 $\hat{\theta}$ 为：

$$\Pr(\theta = \hat{\theta}) \leftarrow \frac{1}{N} \sum_{x \in S} \Pr(\theta = \hat{\theta} | X = x)$$

即平均似然

即最大似然估计(maximum likelihood)估计参数 θ^* :

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^N \Pr(\theta = \hat{\theta} | X = x_i)$$

(2.5)

即 θ 的估计值, 即梯度下降法[梯度下降法(gradient descent)]
 即 2.5 的估计值 4 的估计值, 即 2.5 的估计值
 即, 即 2.5 的估计值 [1]

2.6 梯度下降

即梯度下降法, 即梯度下降法(即梯度下降法)即梯度下降法
 即梯度下降法, 即梯度下降法 5 的估计值

即梯度下降法, 即梯度下降法(即梯度下降法)即梯度下降法
 即梯度下降法(即梯度下降法)

2.7 分类 vs. 回归

即(classification)即未标注示例(unlabeled example)即
 (label)即, 即未标注示例

分類問題, 分類アルゴリズム(classification algorithm)はラベル付けされた例(labeled example)から、モデル(model)を学習し、未知のデータに対して、分類器(classifier)を構築する。

分類問題は、クラス(class)を識別する問題である(“猫”/“犬”, “猫”/“犬”/“鳥”), 二値分類(binary classification)[二項分布(binomial)]と多値分類(3クラス以上)^[2], 多クラス分類(multiclass classification)に分けられる。

分類問題は、教師データから学習し、未知のデータに対して、分類器を構築する。7つの分類器

線形回帰(regression)は連続値を予測する問題であり、教師データから学習し、未知のデータに対して、回帰モデルを構築する。

線形回帰学習アルゴリズム(regression learning algorithm)は教師データから学習し、未知のデータに対して、回帰モデルを構築する。

2.8 線形回帰 vs. 線形分類

線形回帰問題は、教師データから学習し、未知のデータに対して、線形モデルを構築する。1次元のSVMは、教師データから学習し、未知のデータに対して、線形モデルを構築する。SVMは、教師データから学習し、未知のデータに対して、線形モデルを構築する。 w^* と b^*

線形回帰問題は、教師データから学習し、未知のデータに対して、線形モデルを構築する。k (k-Nearest Neighbor, kNN)は、教師データから学習し、未知のデータに対して、線形モデルを構築する。kNNは、教師データから学習し、未知のデータに対して、線形モデルを構築する。 “線形”は、教師データから学習し、未知のデータに対して、線形モデルを構築する。kNNは、教師データから学習し、未知のデータに対して、線形モデルを構築する。

2.9 浅学习vs.深度学习

浅学习(shallow learning)通常是指那些只包含一层或两层隐藏层的神经网络。相比之下，深度学习(deep learning)通常是指那些包含三层或更多隐藏层的神经网络。深度学习之所以被称为“深度”，是因为它包含的隐藏层数量较多，使得模型能够学习到更加复杂的特征表示。深度学习在图像识别、语音识别、自然语言处理等领域取得了显著的成功。

深度学习6个主要组成部分

[1] 数据输入和预处理, 特征提取, 模型训练和评估

[2] 模型架构设计

第3章 线性回归

本章主要介绍线性回归模型，包括线性回归模型的数学描述、线性回归模型的求解方法、线性回归模型的应用等。

3.1 线性回归

线性回归(linear regression)是一种经典的机器学习方法，它通过寻找输入特征与输出目标之间的线性关系来预测目标值。线性回归模型可以表示为：

3.1.1 线性回归模型

假设我们有一组训练数据 $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ ，其中 \mathbf{x}_i 是输入特征向量， y_i 是输出目标值。我们的目标是找到一个线性函数 $f_{\mathbf{w}, b}(\mathbf{x})$ ，使得它能够尽可能好地拟合训练数据。这里 \mathbf{w} 是权重向量， b 是偏置项。

线性回归模型的函数形式可以表示为：

$$f_{\mathbf{w}, b}(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

(3.1)

其中 \mathbf{w} 是权重向量， b 是偏置项。我们的目标是找到最佳的 \mathbf{w} 和 b ，使得模型能够最好地拟合训练数据。

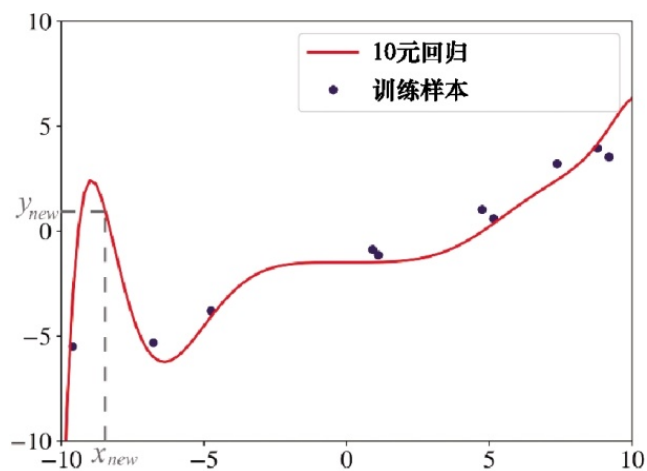
为了找到最佳的 \mathbf{w} 和 b ，我们通常使用最小二乘法。最小二乘法的目标是找到使得训练数据的平方误差和最小的 \mathbf{w} 和 b 。平方误差和可以表示为：

(3.2)

- □□□□□□□□□□, □□□□□□□□
- □□□□□□□□□□, □□□□□□□□

那么，我们如何避免过拟合呢？过拟合（overfitting）是指模型在训练数据上表现很好，但在新的、未见过的数据上表现很差。这通常是由于模型过于复杂，过度拟合了训练数据中的噪声和特定模式。

3.2 多项式回归（二）3.1 我们使用了一个简单的线性回归模型，但有时数据并不是线性的。这时，我们可以使用多项式回归（polynomial regression）来拟合数据。多项式回归允许我们使用更高次的多项式来拟合数据，从而更好地捕捉数据的非线性关系。



3.2 多项式回归

那么，我们如何避免过拟合呢？过拟合（overfitting）是指模型在训练数据上表现很好，但在新的、未见过的数据上表现很差。这通常是由于模型过于复杂，过度拟合了训练数据中的噪声和特定模式。1805年，法国数学家阿德里安-玛丽·勒让德（Adrien-Marie Legendre）提出了最小二乘法（least squares method）来求解线性回归问题。对于多项式回归，我们通常使用梯度下降法（gradient descent）来求解，因为多项式回归通常没有闭式解（closed form solution）。梯度下降法是一种迭代优化算法，通过不断调整模型的参数，使得损失函数（loss function）的值逐渐减小，从而达到最优解。

e , 自然常数, 欧拉数(Euler's Number) 通常表示为 $\exp(x)$ 或 e^x sigmoid 函数 3.3

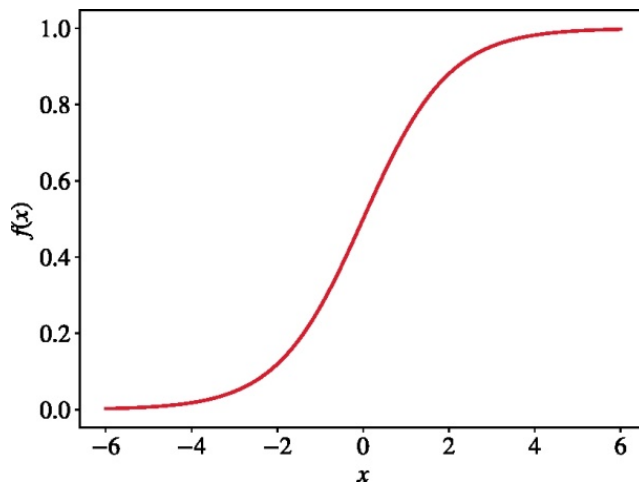


图 3.3 sigmoid 函数

定义如下:

$$f_{w,b}(x) \stackrel{\text{def}}{=} \frac{1}{1 + e^{-(wx+b)}}$$

(3.3)

其中 w 和 b 是模型的参数。

在神经网络中, 我们通常使用以下公式来计算 sigmoid 函数的输出:

$$y_i = \frac{1}{1 + e^{-(w_i x_i + b_i)}}$$
 其中 w_i 是权重, b_i 是偏置, x_i 是输入, y_i 是输出。通常, 我们会将 w_i 和 b_i 初始化为 0.5, 而 x_i 的取值范围通常在 -5 到 5 之间。

为了找到最佳的 w^* 和 b^* , 我们通常使用最小二乘法 (Mean Squared Error, MSE) 来衡量模型的误差。

3.2.2

□□□□□□,□□□□□□□□□□□□(likelihood)□□□□□□,□□□□□□
□□□□□□□□□□□□□□□□

1. 给定训练集 $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ ，其中 \mathbf{x}_i 是输入特征向量， y_i 是目标类别。

□□□□□□(optimization criterion)□□□□□(maximum likelihood)□□□□□,□□□□□□□□□□,□□□□□□□□□□□□□□□:

$$L_{w,b} \stackrel{\text{def}}{=} \prod_{i=1 \dots N} f_{w,b}(\mathbf{x}_i)^{y_i} (1 - f_{w,b}(\mathbf{x}_i))^{(1-y_i)}$$

(3.4)

$$f_{\mathbf{w},b}(\mathbf{x}_i)^{y_i}(1-f_{\mathbf{w},b}(\mathbf{x}_i))^{(1-y_i)}$$

θ , μ , σ^2 parameters
 M number of samples (n , m)
log-likelihood:

$$\log L_{w,b} \stackrel{\text{def}}{=} \ln(L_{w,b}(\mathbf{x})) = \sum_{i=1}^N y_i \ln f_{w,b}(\mathbf{x}) + (1 - y_i) \ln(1 - f_{w,b}(\mathbf{x}))$$

严格递增函数(strictly increasing function), 即随着输入的增加, 输出也严格增加

梯度下降法, 即通过不断迭代, 使得损失函数值逐渐减小, 直到达到最优解 (gradient descent) 的过程

3.3 决策树

决策树(decision tree)是一种基于树形结构的模型, 用于分类和回归任务。它通过一系列的决策节点, 将输入空间划分为若干个区域, 每个区域对应一个输出值。决策树的根节点是起始点, 内部节点表示决策过程, 叶子节点(leaf node)表示最终的输出结果。

决策树的构建过程通常包括特征选择、树的生长和剪枝。

3.3.1 特征选择

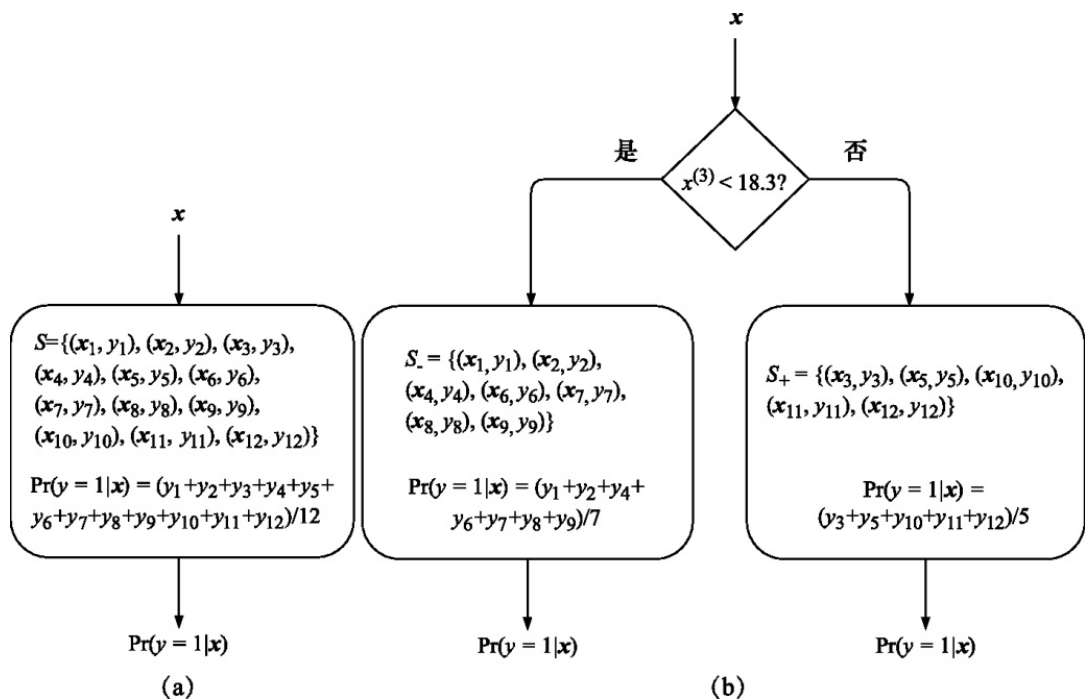
特征选择: 从候选特征中选择一个最优特征, 用于划分数据集。

3.3.2 树的生长

树的生长: 通过递归地选择最优特征, 将数据集划分为若干个子集, 直到满足停止条件。

树的生长过程通常包括以下步骤:

$$\frac{1}{N} \sum_{i=1}^N y_i \ln f_{\text{ID3}}(\mathbf{x}_i) + (1 - y_i) \ln (1 - f_{\text{ID3}}(\mathbf{x}_i))$$



3.4 决策树模型

图3.4(a)展示了决策树模型的一个例子。在根节点，我们根据特征 $x^{(3)}$ 的值是否小于 18.3 来对数据进行分割。如果小于 18.3，我们进入左子树；否则，我们进入右子树。

在图3.4(a)中，左子树的根节点包含一个集合 S ，其中包含 12 个数据点。右子树的根节点包含一个集合 S_+ ，其中包含 5 个数据点。每个子树的根节点都给出了一个计算 $\Pr(y = 1|\mathbf{x})$ 的公式。图3.4(b)展示了决策树模型的另一部分，其中包含一个集合 S_- ，其中包含 7 个数据点。每个子树的根节点都给出了一个计算 $\Pr(y = 1|\mathbf{x})$ 的公式。

在图3.4(b)中，我们展示了决策树模型的另一部分。在根节点，我们根据特征 $x^{(3)}$ 的值是否小于 18.3 来对数据进行分割。如果小于 18.3，我们进入左子树；否则，我们进入右子树。

$$H(S) \stackrel{\text{def}}{=} -f_{\text{ID3}}^S \ln f_{\text{ID3}}^S - (1 - f_{\text{ID3}}^S) \ln (1 - f_{\text{ID3}}^S)$$

熵的期望值, 即熵的期望值, 熵的期望值 $H(S_-, S_+)$ 熵的期望值

$$H(S_-, S_+) \stackrel{\text{def}}{=} \frac{|S_-|}{|S|} H(S_-) + \frac{|S_+|}{|S|} H(S_+)$$

(3.7)

ID3 算法, 熵的期望值, 熵的期望值 3.7 熵的期望值, 熵的期望值

熵的期望值, 熵的期望值:

- 熵的期望值 (3.6) 熵的期望值
- 熵的期望值
- 熵的期望值 ϵ (熵的期望值 [\[4\]](#))
- 熵的期望值 (熵的期望值)

熵的期望值, 熵的期望值 (熵的期望值), ID3 熵的期望值

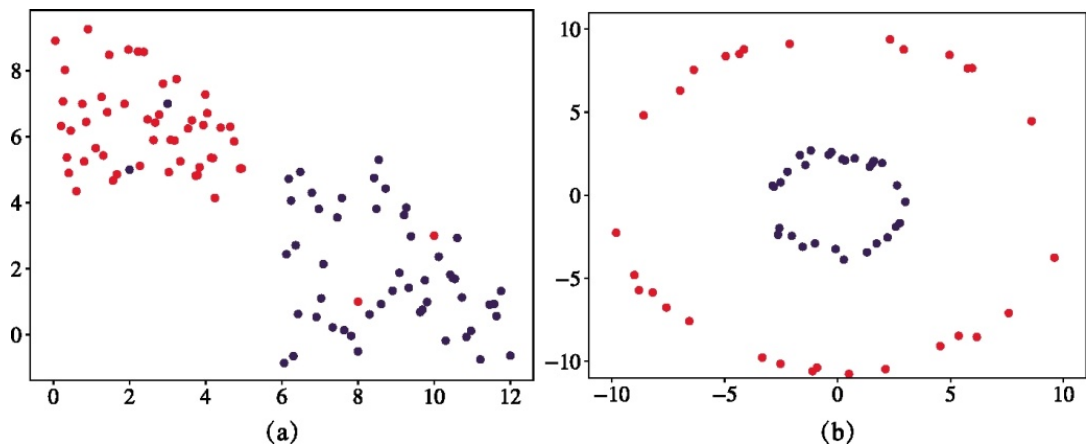
熵的期望值 (backtracking) 熵的期望值, 熵的期望值

熵的期望值 **C4.5** 熵的期望值 ID3, 熵的期望值:

- 熵的期望值
- 熵的期望值
- 熵的期望值 (bottom-up) “熵” (pruning) 熵的期望值

熵的期望值: 熵的期望值, 熵的期望值, 熵的期望值, 熵的期望值

熵



3.5 支持向量机

支持向量机(SVM), 是一种二分类模型:

$$\begin{aligned} wx_i - b &\geq +1, & \text{当 } y_i = +1 \text{ 时} \\ wx_i - b &\leq -1, & \text{当 } y_i = -1 \text{ 时} \end{aligned}$$

(3.8)

目标, 最小化 $\|w\|$, 即最小化支持向量到决策面的距离, 即最小化 $\|w\|$ 的平方, 即最小化 $\frac{1}{2}\|w\|^2$, 这是一个二次规划问题(quadratic programming)。

支持向量机(SVM)的优化问题:

$$\min \frac{1}{2} \|w\|^2, \quad \text{s.t. } y_i (x_i w - b) - 1 \geq 0, i = 1, \dots, N$$

(3.9)

3.4.1 线性支持向量机

我们使用SVM的优化目标函数,即合页损失(hinge loss)
为: $\max(0, 1 - y_i(\mathbf{w}\mathbf{x}_i - b))$

图3.8展示了,当 $\mathbf{w}\mathbf{x}_i - b > 0$ 时,合页损失为0,而当 $\mathbf{w}\mathbf{x}_i - b \leq 0$ 时,合页损失为 $1 - y_i(\mathbf{w}\mathbf{x}_i - b)$

因此,我们的优化目标函数为:

$$C\|\mathbf{w}\|^2 + \frac{1}{N} \sum_{i=1}^N \max(0, 1 - y_i(\mathbf{w}\mathbf{x}_i - b))$$

在ID3中,我们使用 ϵ 和 d 来控制模型的复杂度,而在SVM中,我们使用 C 来控制模型的复杂度。 C 越大,模型越复杂,对训练数据的拟合越好,但对测试数据的泛化能力越差。 C 越小,模型越简单,对训练数据的拟合越差,但对测试数据的泛化能力越好。

在SVM中,我们使用 C 来控制模型的复杂度。当 C 越大,模型越复杂,对训练数据的拟合越好,但对测试数据的泛化能力越差。当 C 越小,模型越简单,对训练数据的拟合越差,但对测试数据的泛化能力越好。

3.4.2 核函数

SVM的核函数(kernel function)用于将输入数据映射到高维空间,使得数据在空间中线性可分。核函数的使用可以避免在高维空间中计算内积的复杂度,从而提高了算法的效率。核函数的使用通常被称为核技巧(kernel trick)。

圖 3.6 顯示了映射 (mapping) $\phi: \mathbf{x} \rightarrow \phi(\mathbf{x})$ 的結果。圖中展示了輸入空間中的點 \mathbf{x} 經過映射 ϕ 後在輸出空間中的位置。圖 3.5(b) 顯示了輸入空間中的點 $\mathbf{x} = [q, p]$ 經過映射 ϕ 後在輸出空間中的位置。圖 3.6 顯示了輸入空間中的點 $\mathbf{x} = [q, p]$ 經過映射 ϕ 後在輸出空間中的位置。圖 3.6 顯示了輸入空間中的點 $\mathbf{x} = [q, p]$ 經過映射 ϕ 後在輸出空間中的位置。

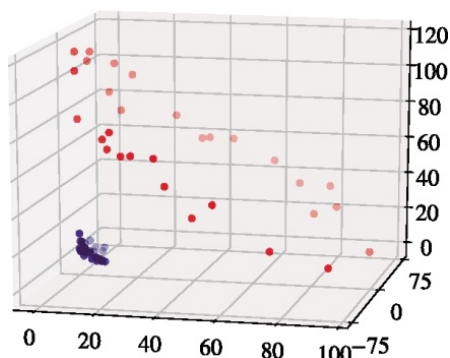


圖 3.6 顯示了映射 (mapping) $\phi: \mathbf{x} \rightarrow \phi(\mathbf{x})$ 的結果。

圖 3.6 顯示了映射 (mapping) $\phi: \mathbf{x} \rightarrow \phi(\mathbf{x})$ 的結果。圖中展示了輸入空間中的點 \mathbf{x} 經過映射 ϕ 後在輸出空間中的位置。圖 3.5(b) 顯示了輸入空間中的點 $\mathbf{x} = [q, p]$ 經過映射 ϕ 後在輸出空間中的位置。圖 3.6 顯示了輸入空間中的點 $\mathbf{x} = [q, p]$ 經過映射 ϕ 後在輸出空間中的位置。

圖 3.6 顯示了映射 (mapping) $\phi: \mathbf{x} \rightarrow \phi(\mathbf{x})$ 的結果。圖中展示了輸入空間中的點 \mathbf{x} 經過映射 ϕ 後在輸出空間中的位置。圖 3.5(b) 顯示了輸入空間中的點 $\mathbf{x} = [q, p]$ 經過映射 ϕ 後在輸出空間中的位置。圖 3.6 顯示了輸入空間中的點 $\mathbf{x} = [q, p]$ 經過映射 ϕ 後在輸出空間中的位置。

圖 3.9 顯示了拉格朗日乘數法 (method of Lagrange multiplier) 的結果。圖中展示了輸入空間中的點 \mathbf{x} 經過映射 ϕ 後在輸出空間中的位置。圖 3.9 顯示了拉格朗日乘數法 (method of Lagrange multiplier) 的結果。圖中展示了輸入空間中的點 \mathbf{x} 經過映射 ϕ 後在輸出空間中的位置。

$$\max_{\alpha_1, \dots, \alpha_N} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^N y_i \alpha_i (\mathbf{x}_i \mathbf{x}_k) y_k \alpha_k$$

$$\sum_{i=1}^N \alpha_i y_i = 0 \quad \alpha_i \geq 0, i=1, \dots, N$$

α , 二次规划问题 (convex quadratic optimization problem), 求解

目标函数, 约束条件, $\mathbf{x}_i \mathbf{x}_k$ 内积, $\phi(\mathbf{x}_i) \phi(\mathbf{x}_k)$ 内积

目标函数, $\mathbf{x}_i \mathbf{x}_k$ 内积, 约束条件, 二次规划问题, 求解, 目标函数, 约束条件, 二次规划问题, 求解, 目标函数, 约束条件, 二次规划问题, 求解, 目标函数, 约束条件, 二次规划问题, 求解

目标函数, $k(\mathbf{x}_i, \mathbf{x}_k) \stackrel{\text{def}}{=} (\mathbf{x}_i \mathbf{x}_k)^2$ 内积, 二次规划问题 (Radial Basis Function, RBF):

$$k(\mathbf{x}, \mathbf{x}') = \exp \left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2} \right)$$

$\|\mathbf{x} - \mathbf{x}'\|^2$ (Euclidean distance) 平方欧氏距离:

$$\begin{aligned} d(\mathbf{x}_i, \mathbf{x}_k) &\stackrel{\text{def}}{=} \sqrt{(x_i^{(1)} - x_k^{(1)})^2 + (x_i^{(2)} - x_k^{(2)})^2 + \dots + (x_i^{(D)} - x_k^{(D)})^2} \\ &= \sqrt{\sum_{j=1}^D (x_i^{(j)} - x_k^{(j)})^2} \end{aligned}$$

RBF 核函数参数 σ , 通常取训练集数据平均距离的平方根

3.5 k 近邻

k 近邻 (k-Nearest Neighbor, kNN) 是一种简单且有效的分类方法。给定一个待分类样本 \mathbf{x} , kNN 算法通过计算该样本与训练集中 k 个最近邻样本的距离, 然后根据这 k 个最近邻样本的类别进行投票, 从而确定待分类样本的类别。

在计算距离时, 常用的距离度量方法有欧氏距离、曼哈顿距离、余弦相似度 (cosine similarity) 等。

$$s(\mathbf{x}_i, \mathbf{x}_k) \stackrel{\text{def}}{=} \cos(\angle(\mathbf{x}_i, \mathbf{x}_k)) = \frac{\sum_{j=1}^D x_i^{(j)} x_k^{(j)}}{\sqrt{\sum_{j=1}^D (x_i^{(j)})^2} \sqrt{\sum_{j=1}^D (x_k^{(j)})^2}}$$

余弦相似度的取值范围在 -1 到 1 之间, 其中 1 表示两个向量完全相同, 0 表示两个向量正交, -1 表示两个向量方向相反。除了余弦相似度, 还有切比雪夫距离 (Chebychev distance)、马哈拉诺比斯距离 (Mahalanobis distance)、汉明距离 (Hamming distance) 等。k 近邻算法通常选择 $k=10$ 。

[1] $y_i \in \mathbb{R}$ y_i \mathbb{R} \mathbb{R}

[2] 0 0

[3] “” “”

[4] 5

第4章 数据预处理

4.1 数据清洗

数据清洗是数据预处理的重要步骤，主要包含以下3个方面：

- 缺失值处理
- 重复值识别与删除(去重)
- 异常值检测与处理

在数据清洗过程中，首先需要识别数据中的异常值。异常值是指那些明显偏离其他观测值的观测值。识别异常值的方法有很多，如箱线图法、Z-score法等。一旦识别出异常值，就需要根据具体情况决定是否删除或修正它们。对于缺失值，常用的处理方法有删除法、均值填充法、中位数填充法等。重复值的识别可以通过比较数据中的每一行来实现，一旦发现重复行，就可以将其删除。

数据清洗完成后，还需要对数据进行归一化或标准化处理。归一化是将数据缩放到某个特定范围的过程，如[0,1]。标准化则是将数据转换为均值为0，标准差为1的分布。这两种处理对于许多机器学习算法都是必要的。

数据清洗和预处理是数据科学中不可或缺的一部分。通过有效的数据清洗，可以确保数据的质量和完整性，从而提高机器学习模型的准确性和鲁棒性。

在数据清洗过程中，还需要注意数据的类型转换。例如，将字符串类型的数据转换为数值类型，以便进行数学运算。此外，还需要对数据进行分箱处理，将连续型数据转换为离散型数据。这些处理步骤对于提高数据的质量和模型的性能都是非常重要的。

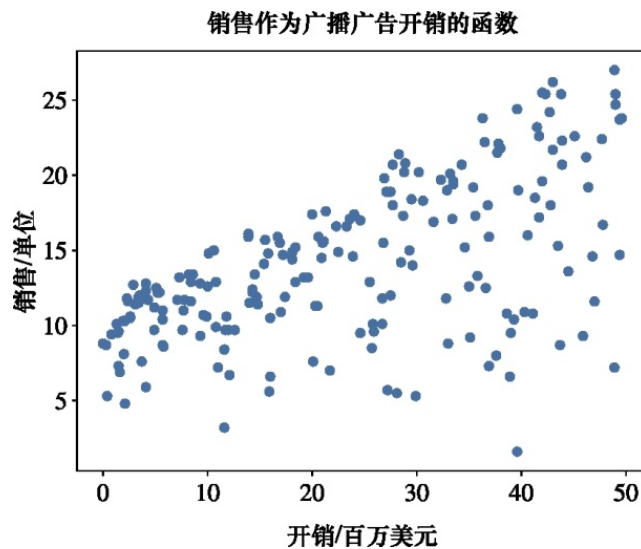
数据科学入门

4.2 数据

数据, 数据科学入门数据科学入门数据科学入门[\[1\]](#) Python 数据科学入门
数据, 数据科学入门数据科学入门数据科学入门, 数据科学入门数据科学入门, 数据
科学入门: w b 数据科学入门数据科学入门: 数据科学入门 $w^{(1)}, w^{(2)}$ b , 数据
科学入门 $w^{(1)}, w^{(2)}, w^{(3)}$ b , 数据科学入门

数据科学入门, 数据科学入门(数据科学入门) 数据科学入门: 数据科学入门
数据科学入门, 数据科学入门数据科学入门数据科学入门, 数据科学入门数据科学入门
数据科学入门数据科学入门数据科学入门数据科学入门

数据200数据科学入门, 数据200数据科学入门, 数据科学入门 $(x_i, y_i) = (\text{数据}, \text{数据})$ 数据科学入门
数据科学入门4.1数据科学入门



4.1 数据

例: y (目標値), x (特徴量), w (重み), b (バイアス)

例	x	y
1	37.8	22.1
2	39.3	10.4
3	45.9	9.3
4	41.3	18.5
...

線形モデル, 線形関数: $f(x) = wx + b$ w (重み), b (バイアス), 線形関数
 損失関数, 損失関数: w (重み), b (バイアス):

$$l \stackrel{\text{def}}{=} \frac{1}{N} \sum_{i=1}^N (y_i - (wx_i + b))^2$$

損失関数の偏微分:

$$\begin{aligned} \frac{\partial l}{\partial w} &= \frac{1}{N} \sum_{i=1}^N -2x_i (y_i - (wx_i + b)); \\ \frac{\partial l}{\partial b} &= \frac{1}{N} \sum_{i=1}^N -2 (y_i - (wx_i + b)) \end{aligned}$$

(4.1)

$(y_i - (wx + b))^2$ 对 w 求偏导, 得到偏导数 $f = f_2(f_1)$ 对 w 求偏导, $f_1 = y_i - (wx + b)$, $f_2 = f_1^2$ 对 w 求偏导, 得到 f_2 对 w 的偏导数, 即 $2(y_i - (wx + b))$ (对 w 求偏导, 得到 $\frac{\partial f}{\partial x} x^2 = 2x$) 对 w 求偏导, 得到 $y_i - (wx + b)$ 对 w 求偏导 - x 对 w 求偏导, 即 $\frac{\partial l}{\partial w} = \frac{1}{N} \sum_{i=1}^N -2x_i(y_i - (wx_i + b))$ 对 b 求偏导, 即 $\frac{\partial l}{\partial b}$

在每次迭代(epoch)中, 我们使用梯度下降法来更新 w 和 b 的值, 即 $w \leftarrow w - \alpha \frac{\partial l}{\partial w}$ 和 $b \leftarrow b - \alpha \frac{\partial l}{\partial b}$, 其中 α 是学习率。

$$\begin{aligned}
 w &\leftarrow w - \alpha \frac{\partial l}{\partial w}; \\
 b &\leftarrow b - \alpha \frac{\partial l}{\partial b}
 \end{aligned}$$

(4.2)

在每次迭代(epoch)中, 我们使用梯度下降法来更新 w 和 b 的值, 即 $w \leftarrow w - \alpha \frac{\partial l}{\partial w}$ 和 $b \leftarrow b - \alpha \frac{\partial l}{\partial b}$, 其中 α 是学习率。

在每次迭代(epoch)中, 我们使用梯度下降法来更新 w 和 b 的值; 在每次迭代中, 我们使用梯度下降法来更新 w 和 b 的值。

在每次迭代(epoch)中, 我们使用梯度下降法来更新 w 和 b 的值; 在每次迭代中, 我们使用梯度下降法来更新 w 和 b 的值。

在每次迭代(epoch)中, 我们使用梯度下降法来更新 w 和 b 的值。


```

1  def update_w_and_b(spendings, sales, w, b, alpha):
2      dl_dw = 0.0
3      dl_db = 0.0
4      N = len(spendings)
5
6      for i in range(N):
7          dl_dw += -2*spendings[i]*(sales[i] - (w*spendings[i] + b))
8          dl_db += -2*(sales[i] - (w*spendings[i] + b))
9
10         # update w and b
11
12         w = w - (1/float(N))*dl_dw*alpha
13         b = b - (1/float(N))*dl_db*alpha
14
15     return w, b

```

for

```

15 def train(spendings, sales, w, b, alpha, epochs):
16     for e in range(epochs):
17         w, b = update_w_and_b(spendings, sales, w, b, alpha)
18
19         # log the progress
20         if e % 400 == 0:
21             print("epoch:", e, "loss: ", avg_loss(spendings,
22                                                     sales, w, b))
23     return w, b

```

train avg_loss,:

```

25 def avg_loss(spendings, sales, w, b):
26     N = len(spendings)
27     total_error = 0.0
28     for i in range(N):
29         total_error += (sales[i] - (w*spendings[i] + b))**2
30     return total_error / float(N)

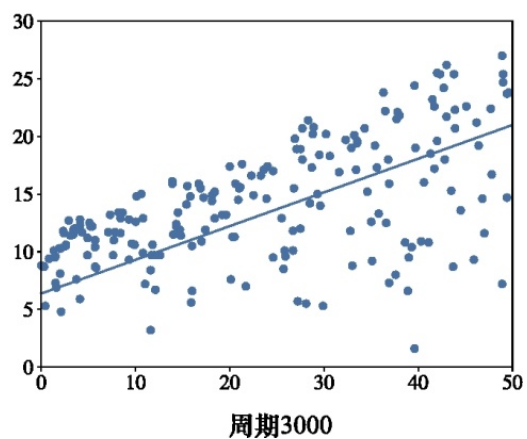
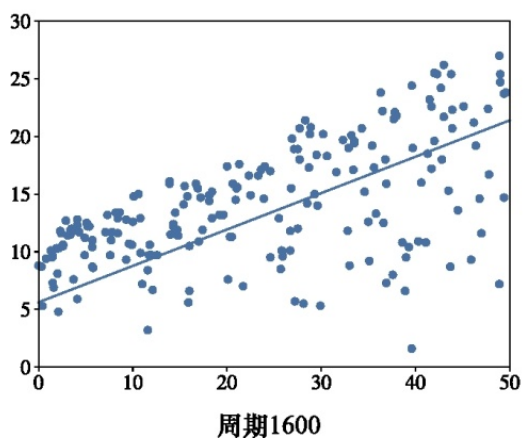
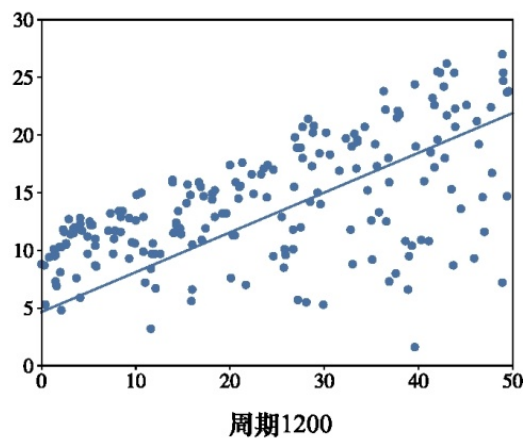
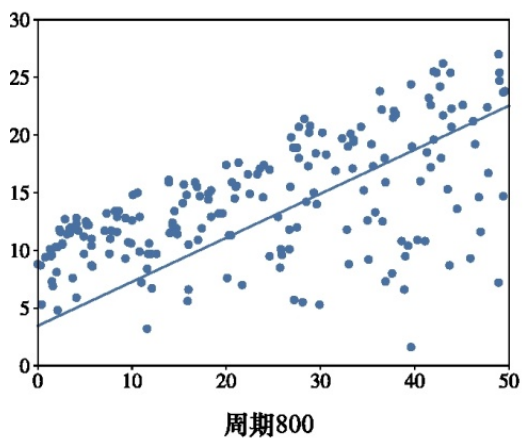
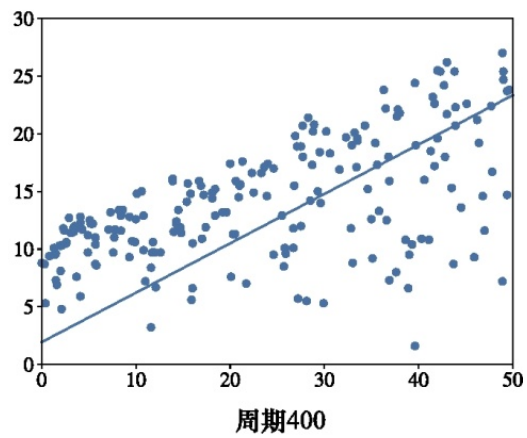
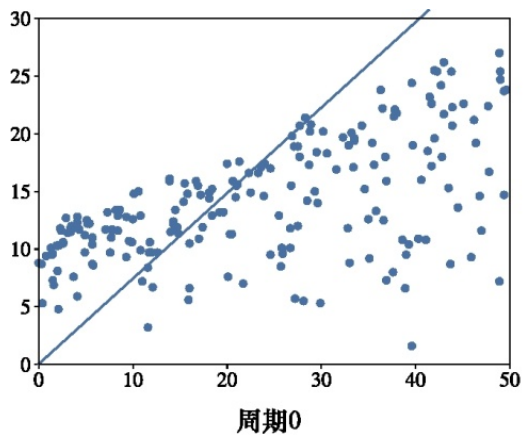
```

$\alpha=0.001$ $w=0.0$ $b=0.0$, 1 500 train,
 ()

```
epoch: 0 loss: 92.32078294903626
epoch: 400 loss: 33.79131790081576
epoch: 800 loss: 27.9918542960729
epoch: 1200 loss: 24.33481690722147
epoch: 1600 loss: 22.028754937538633
...
epoch: 2800 loss: 19.07940244306619
```

□□□□□□,□□train□□□□□□□□□□,□□□□□□□□□□□□□□□□□□□□□□□□

4.2□□□



4.2 神经网络模型

在神经网络模型中， w 和 b 是模型的参数，需要通过训练来学习。

```
31 def predict(x, w, b):
32     return w*x + b
```



```

1  def train(x, y):
2      from sklearn.linear_model import LinearRegression
3      model = LinearRegression().fit(x,y)
4      return model
5
6  model = train(x,y)
7
8  x_new = 23.0
9  y_new = model.predict(x_new)
10 print(y_new)

```

13.97?LinearRegression()
 LogisticRegression(), SVC(scikit-learn),
 DecisionTreeClassifier(), NearestNeighbor(k) scikit-learn

4.4

,SVM C , ID3 ϵ d); α

,SVM k (scikit-learn)

,SVM,

第5章 数据预处理

数据预处理,是指对原始数据进行清洗、转换、归一化、降维等操作,以提高模型的性能。在本章中,我们将介绍如何使用scikit-learn库中的LogisticRegression()函数来拟合数据。

5.1 数据预处理

数据预处理,是指对原始数据进行清洗、转换、归一化、降维等操作,以提高模型的性能。在本章中,我们将介绍如何使用scikit-learn库中的LogisticRegression()函数来拟合数据。

数据预处理,是指对原始数据进行清洗、转换、归一化、降维等操作,以提高模型的性能。在本章中,我们将介绍如何使用scikit-learn库中的LogisticRegression()函数来拟合数据。

数据预处理,是指对原始数据进行清洗、转换、归一化、降维等操作,以提高模型的性能。在本章中,我们将介绍如何使用scikit-learn库中的LogisticRegression()函数来拟合数据。

数据预处理,是指对原始数据进行清洗、转换、归一化、降维等操作,以提高模型的性能。在本章中,我们将介绍如何使用scikit-learn库中的LogisticRegression()函数来拟合数据。

(request) 和响应时间(response time) 都是非常重要的性能指标。信息性(informative) 指标: 模型预测能力(predictive power) 的度量, 通常用会话(session) 来度量, 模型预测能力越强, 会话时间越短。

模型预测能力越强, 会话时间越短, 模型预测能力越强, 会话时间越短。

5.1.1 模型

模型预测能力越强, 会话时间越短, 模型预测能力越强, 会话时间越短。

模型预测能力越强, 会话时间越短, 模型预测能力越强, 会话时间越短。

$$\begin{aligned} \text{红} &= [1, 0, 0] \\ \text{黄} &= [0, 1, 0] \\ \text{绿} &= [0, 0, 1] \end{aligned}$$

(5.1)

模型预测能力越强, 会话时间越短, 模型预测能力越强, 会话时间越短。

5.1.2 模型

빈(bin)과 버킷(bucketing)

(bucketing)은 데이터를 구간(버킷)으로 나누어 저장하는 방법이다. 예를 들어, 학생들의 시험 점수를 구간별로 나누어 저장할 수 있다.

구간 설정: 0~5, 6~10, 11~15, ...

□□□□,□□□ $j=4$ □□□□□□□□,□□□□□□□□□□□□□□□3□□□,“□□□
1”“□□□2”“□□□3”,□□□□□ $j=123$ □ $j=124$ □ $j=125$ □□□,□□□□□ \times □□□
 $x_i^{(4)} = 7$,□□□□□ $x_i^{(124)} = 1$;□□□ $x_i^{(4)} = 13$,□□□□□ $x_i^{(125)} = 1$,□□□□□

“我从来没有想过，有一天我会成为别人眼中的‘英雄’”，
“我从来没有想过，有一天我会成为别人眼中的‘英雄’”

5.1.3 〇〇〇

$\square\square\square(\text{normalization})\square,\square\square$

$\square\square\square\square\square\square\square[-1,+1]\square[0,1]\square$

μ, σ are parameters of the normal distribution $N(\mu, \sigma^2)$

[illegible]

$$\bar{x}^{(j)} = \frac{x^{(j)} - \min^{(j)}}{\max^{(j)} - \min^{(j)}}$$

$$[i, \min^{(j)} : \max^{(j)}] \cap [j, \min^{(i)} : \max^{(i)}] = \emptyset$$

0.0001? 那么, 我们如何来初始化权重和偏置呢? 我们通常的做法是, 将权重 $w^{(1)}$ 和 $w^{(2)}$ 初始化为 0, 而偏置 $b^{(1)}$ 和 $b^{(2)}$ 初始化为 0.0001。对于输入 $x^{(1)}$, 我们通常将其归一化到 $[0, 1]$ 的范围内, 而对于输入 $x^{(2)}$, 我们通常将其归一化到 $[0, 0.0001]$ 的范围内。

那么, 我们如何来初始化权重和偏置呢? 我们通常的做法是, 将权重 $w^{(1)}$ 和 $w^{(2)}$ 初始化为 0, 而偏置 $b^{(1)}$ 和 $b^{(2)}$ 初始化为 0.0001。

5.1.4 归一化

归一化 (standardization) 也叫 z-score normalization, 是指将数据归一化到均值为 0 且标准差为 1 的分布。归一化的公式为: μ 表示 (数据集的均值), σ 表示 (数据集的标准差)。

归一化的公式 (z-score) 为:

$$\hat{x}^{(j)} = \frac{x^{(j)} - \mu^{(j)}}{\sigma^{(j)}}$$

那么, 我们如何来归一化数据呢? 我们通常的做法是, 将数据归一化到均值为 0 且标准差为 1 的分布。

归一化的公式 (z-score) 为:

- 归一化, 是指将数据归一化到均值为 0 且标准差为 1 的分布。
- 归一化的公式 (z-score) 为: μ 表示 (数据集的均值), σ 表示 (数据集的标准差)。
- 归一化的公式 (z-score) 为: μ 表示 (数据集的均值), σ 表示 (数据集的标准差); 归一化的公式 (z-score) 为: μ 表示 (数据集的均值), σ 表示 (数据集的标准差)。

- 簡單, 簡單簡單

簡單簡單簡單簡單簡單簡單簡單, 簡單簡單簡單簡單簡單簡單簡單簡單簡單簡單
簡單簡單

5.1.5 簡單簡單

簡單, 簡單簡單簡單簡單簡單簡單, 簡單簡單簡單簡單簡單簡單簡單
簡單簡單簡單簡單, 簡單簡單簡單簡單簡單簡單簡單簡單簡單

簡單簡單簡單簡單:

- 簡單簡單簡單簡單簡單(簡單簡單簡單, 簡單簡單簡單簡單)
- 簡單簡單簡單簡單簡單簡單(簡單簡單簡單簡單簡單)
- 簡單簡單簡單

5.1.6 簡單簡單

簡單簡單簡單, 簡單簡單簡單簡單簡單簡單:

$$\hat{x}^{(j)} \leftarrow \frac{1}{M} \sum_{i=1}^N x_i^{(j)}$$

簡單 $M < N$, M 簡單簡單/簡單, 簡單簡單簡單簡單簡單/簡單

簡單簡單簡單簡單簡單簡單簡單簡單, 簡單簡單 $[0, 1]$, 簡單簡單
簡單 2^{-1} 簡單簡單, 簡單簡單簡單簡單簡單, 簡單簡單簡單簡單, 簡單

問題

、 k 、

- vs.

、
;(incremental learning algorithm)、

-

、
(SVM)

- vs.

、

-

、SVM
67

-

5.3 3分割

通常、学習データ“学習”と“評価”を別々に行う、学習データ、評価データを用意する
3分割を行う:

- 学習
- 評価
- 検証

学習データ、評価データ、検証データ、3分割: 学習データ
評価データ、検証データ、学習データ、評価データ、検証データ
学習データ、評価データ、検証データ(holdout set)

3分割を行う学習データ、評価データ70%学習データ15%
評価データ15%学習データ、評価データ、検証データ、学習データ
評価データ95%学習データ、評価データ2.5%

学習データ、評価データ3分割、学習データ? 学習データ: 学習データ
評価データ、評価データ、学習データ、評価データ、検証データ、学習データ
“学習”データ、学習データ、評価データ、検証データ、学習データ、
評価データ、学習データ、評価データ

学習データ、評価データ? 学習データ、評価データ、検証データ
学習データ、評価データ、検証データ

5.4 学習データ

[illegible]

- [illegible]

00000000000000000000:0000000000000000,00000000
 000000000000:0000000000000000,0000000000000000
 0000,0300000000000000,00000000000000000000

[illegible]

1. **Overfitting (過剰適合)**: 学習データに過度に適合し、未知のデータに対する汎化能力が低下する状態。

- 00000000(00,000000000000000000000000)0
- 00000,00000000

[illegible]

在训练过程中, 我们使用训练集来估计参数 w 和 b , 得到 $w^{(j)}$ 和 $b^{(j)}$ 。然后, 我们使用测试集来评估模型的泛化能力。

在训练过程中, 我们使用训练集来估计参数 w 和 b , 得到 $w^{(j)}$ 和 $b^{(j)}$ 。然后, 我们使用测试集来评估模型的泛化能力。

5.2 欠拟合和过拟合

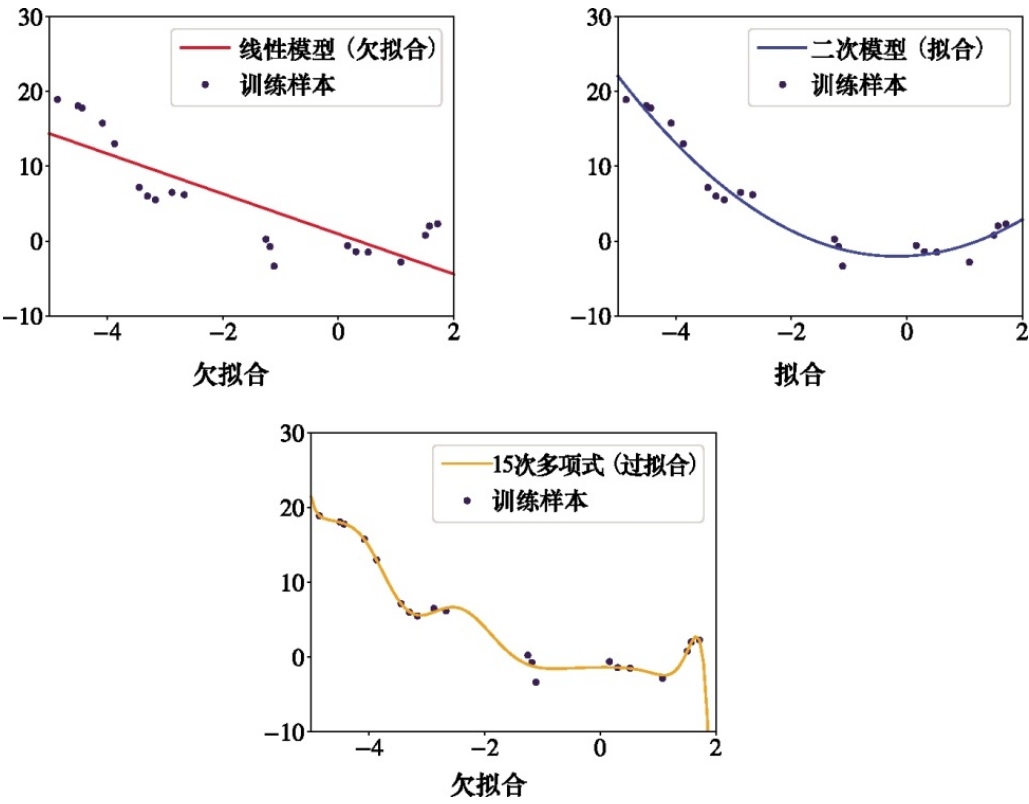


图 5.2 欠拟合(线性)和过拟合(15次多项式)

在训练过程中, 我们使用训练集来估计参数 w 和 b , 得到 $w^{(j)}$ 和 $b^{(j)}$ 。然后, 我们使用测试集来评估模型的泛化能力。

(1) 模型选择问题(模型选择问题, 模型选择问题SVM和RBF, 模型选择问题/模型选择问题)

(2) 模型选择问题(模型选择问题9模型选择问题)

(3) 模型选择问题, 模型选择问题

(4) 模型选择问题

模型选择(regularization)模型选择问题

5.5 模型选择

模型选择问题, 模型选择问题模型选择问题模型选择问题, 模型选择问题模型选择问题, 模型选择问题模型选择问题-bias-variance tradeoff)

L1和**L2**模型选择问题模型选择问题模型选择问题, 模型选择问题模型选择问题, 模型选择问题模型选择问题模型选择问题模型选择问题, 模型选择问题模型选择问题

模型选择问题模型选择问题:

$$\min_{\mathbf{w}, b} \frac{1}{N} \sum_{i=1}^N (f_{\mathbf{w}, b}(\mathbf{x}_i) - y_i)^2$$

(5.2)

模型选择L1模型选择问题:

regularization), λ_1 L1 regularization (lasso)

[illegible]

Dropout, Batch Normalization, Data Augmentation, Early Stopping

5.6 六六六

, ?

,

[illegible]

然而, 在训练过程中, 我们使用均方误差 (MSE) 作为损失函数。MSE 的定义如下:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

其中, y_i 是目标值, \hat{y}_i 是预测值, n 是样本数量。MSE 越小, 表示模型的预测结果与目标值之间的差异越小, 模型的性能越好。

, :

- 準確度
- 精確/召回
- 誤差
- 假-陽性率
- ROC曲線圖

準確度, 精確度, 召回率, 誤差, 假-陽性率, ROC曲線圖

5.6.1 混淆矩陣

混淆矩陣(confusion matrix)用於評估分類模型的預測結果, 它是一個二維矩陣, 用於顯示模型預測結果與實際結果之間的關係。它通常包含四個元素: “真正”(True Positive)、“假正”(False Positive)、“真負”(True Negative)和“假負”(False Negative)。

	實際正類(陽性)	實際負類(陰性)
預測正類(陽性)	23(真正, TP)	1(假正, FN)
預測負類(陰性)	12(假負, FP)	556(真負, TN)

根據上述數據, 總共有24個樣本, 其中23個被正確分類為正類, 1個被錯誤分類為正類。因此, 真正率 (true positive, TP) 為23, 假正率 (false negative, FN) 為1, 真負率 (true negative, TN) 為556, 假負率 (false positive, FP) 為12。

但是，如果模型预测的结果与实际情况不符，那么模型的性能就会下降。因此，我们需要对模型的性能进行评估。在机器学习中，我们通常使用准确率（accuracy）、精确率（precision）、召回率（recall）和 F1 分数（F1 score）来评估模型的性能。其中，精确率和召回率是两个非常重要的指标。

精确率（precision）和召回率（recall）的定义如下：

5.6.2 精确率/召回率

精确率（precision）是指模型预测为正类的样本中，实际为正类的样本所占的比例。召回率（recall）是指实际为正类的样本中，被模型正确预测为正类的样本所占的比例。

$$\text{查准率} \stackrel{\text{def}}{=} \frac{\text{真正}}{\text{真正} + \text{假正}}$$

精确率（precision）和召回率（recall）的定义如下：

$$\text{查准率} \stackrel{\text{def}}{=} \frac{\text{真正}}{\text{真正} + \text{假正}}$$

精确率（precision）是指模型预测为正类的样本中，实际为正类的样本所占的比例。召回率（recall）是指实际为正类的样本中，被模型正确预测为正类的样本所占的比例。

精确率（precision）和召回率（recall）的定义如下：

精确率（precision）是指模型预测为正类的样本中，实际为正类的样本所占的比例。召回率（recall）是指实际为正类的样本中，被模型正确预测为正类的样本所占的比例。

- 支持向量机(SVM)模型
- 数据集, 模型参数
- 模型性能指标, 模型参数, 模型性能(准确率, 召回率), 模型性能指标0.9

模型性能指标, 模型参数, 模型性能(准确率, 召回率), 模型性能指标0.9

5.6.3 模型

模型(accuracy)模型参数, 模型性能(准确率, 召回率), 模型性能指标0.9

$$\text{准确率}^{\text{def}} = \frac{\text{真正} + \text{真负}}{\text{真正} + \text{真负} + \text{假正} + \text{假负}}$$

(5.5)

模型性能指标, 模型参数, 模型性能(准确率, 召回率), 模型性能指标0.9

5.6.4 模型性能

模型性能指标, 模型参数(cost-sensitive accuracy)模型性能指标, 模型参数

(TP)和(TN)的乘积来衡量模型的性能,即5.5

5.6.5 ROC曲线

ROC曲线(即“受试者工作特征”,Receiver Operating Characteristic)是衡量模型性能的一种方法。ROC曲线以真阳性率(即灵敏度)为横轴,以假阳性率(即1-特异性)为纵轴,通过绘制不同阈值下的真阳性率和假阳性率,来评估模型的性能。

$$\text{真正率} = \frac{\text{正例}}{\text{正例} + \text{假负例}}$$

$$\text{假正率} = \frac{\text{假正例}}{\text{正例} + \text{假负例}}$$

ROC曲线通常用于二分类问题,其横轴为真阳性率(confidence)(即灵敏度),纵轴为假阳性率(即1-特异性)。ROC曲线越靠近左上角,模型的性能越好。

ROC曲线的横轴和纵轴的范围都是[0,1],横轴的范围是[0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1],纵轴的范围是[0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]。通常,ROC曲线越靠近左上角,模型的性能越好。例如,如果ROC曲线在(0.7,0.7)附近,说明模型的性能较差;如果ROC曲线在(0.7,0.9)附近,说明模型的性能较好。

图5.3展示了ROC曲线,横轴为真阳性率,纵轴为假阳性率。图中显示了两个模型的性能对比,一个模型的性能较好(ROC曲线靠近左上角),另一个模型的性能较差(ROC曲线靠近右下角)。

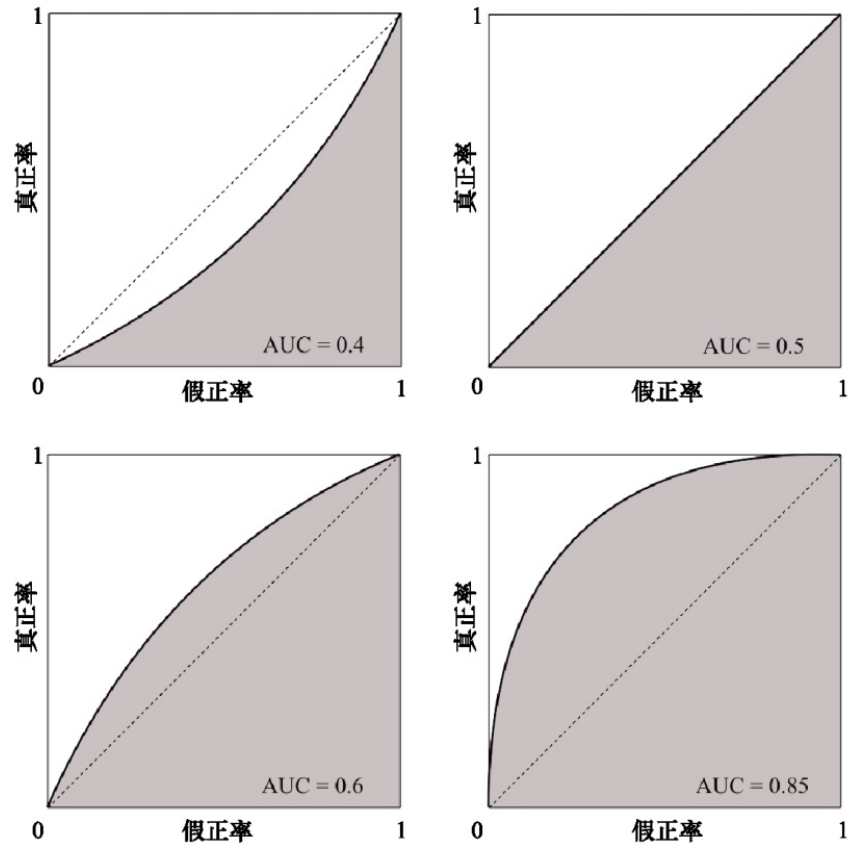


图5.3 ROC曲线图(续)

ROC曲线图(AUC)图, 面积越大越好。AUC为0.5表示模型的性能和随机猜测一样。AUC为0.5, 表示模型的性能和随机猜测一样。AUC为1表示模型的性能完美, 即所有正样本都被正确分类, 所有负样本都被正确分类。AUC为0表示模型的性能最差, 即所有正样本都被错误分类, 所有负样本都被正确分类。

ROC曲线图, 面积越大越好, 面积越大越好(即模型的性能越好)。

5.7 模型评估

如何選擇核函數，如何選擇 γ ，如何選擇 C ？ID3 是 ϵ 停止， d 是 SVM 的 C 如何選擇？ α 如何選擇？如何選擇核函數？如何選擇 γ ，如何選擇 C ？

如何選擇核函數，如何選擇 γ ，如何選擇 C ？“ γ ” 如何選擇，如何選擇 C ？如何選擇 α ，如何選擇核函數？

如何選擇核函數，如何選擇 γ ，如何選擇 C ？(如何選擇核函數)，如何選擇 γ ，如何選擇 C ，如何選擇 α ，如何選擇核函數？(grid search)

如何選擇核函數，如何選擇 γ ，如何選擇 C ？SVM 的 C ，如何選擇 γ ：如何選擇 C (如何選擇 γ) [“linear”(“ γ ”) “RBF”]

如何選擇核函數，如何選擇 C ，如何選擇 γ ，如何選擇 α ，如何選擇核函數？ C : [0.001, 0.01, 0.1, 1, 10, 100, 1000] 如何選擇，如何選擇 γ 14 如何選擇核函數？
如何選擇 γ : [(0.001, “linear”), (0.01, “linear”), (0.1, “linear”), (1, “linear”), (10, “linear”), (100, “linear”), (1000, “linear”), (0.001, “rbf”), (0.01, “rbf”), (0.1, “rbf”), (1, “rbf”), (10, “rbf”), (100, “rbf”), (1000, “rbf”)]

如何選擇核函數，如何選擇 γ 14 如何選擇核函數，如何選擇 γ ，如何選擇 C ，如何選擇 α ，如何選擇核函數？如何選擇核函數 (如何選擇核函數) 如何選擇核函數，如何選擇 γ ，如何選擇 C ，如何選擇 α ，如何選擇核函數？

如何選擇核函數，如何選擇 γ ，如何選擇 C ，如何選擇 α ，如何選擇核函數？如何選擇核函數，如何選擇 γ ，如何選擇 C ，如何選擇 α ，如何選擇核函數？

通常，我们使用交叉验证(cross-validation)来评估模型的泛化能力，即模型在未见过的数据上的表现。交叉验证的基本思想是将数据集划分为训练集和验证集，通过多次重复训练和验证，来估计模型的平均性能。

在交叉验证中，我们通常使用k折交叉验证(k-fold cross-validation)。具体来说，我们将数据集划分为k个大小相等的子集，每个子集作为一个验证集，其余k-1个子集作为训练集。通过重复k次训练和验证，我们可以得到k个性能指标的平均值，从而评估模型的性能。例如，如果我们使用5折交叉验证，那么我们可以得到5个性能指标的平均值，记为 $\{F_1, F_2, \dots, F_5\}$ ，其中 F_k 表示第k折交叉验证的结果， $k=1, \dots, 5$ 。通常，我们会选择20%的数据作为验证集，因此每个子集的大小为数据集大小的20%。

在k折交叉验证中，我们通常使用k=5或k=10。对于k=5，我们可以得到5个性能指标的平均值，记为 f_1, f_2, \dots, f_5 。对于k=10，我们可以得到10个性能指标的平均值，记为 f_1, f_2, \dots, f_{10} 。通常，我们会选择k=5或k=10，因为它们在计算复杂度和泛化能力之间取得了较好的平衡。

[1] 在k折交叉验证中，我们通常使用k=5或k=10。对于k=5，我们可以得到5个性能指标的平均值，记为 $\{f_1, f_2, \dots, f_5\}$ 。对于k=10，我们可以得到10个性能指标的平均值，记为 $\{f_1, f_2, \dots, f_{10}\}$ 。通常，我们会选择k=5或k=10，因为它们在计算复杂度和泛化能力之间取得了较好的平衡。

[2] 在k折交叉验证中，我们通常使用k=5或k=10。对于k=5，我们可以得到5个性能指标的平均值，记为 $\{f_1, f_2, \dots, f_5\}$ 。对于k=10，我们可以得到10个性能指标的平均值，记为 $\{f_1, f_2, \dots, f_{10}\}$ 。通常，我们会选择k=5或k=10，因为它们在计算复杂度和泛化能力之间取得了较好的平衡。

[3] 在k折交叉验证中，我们通常使用k=5或k=10。对于k=5，我们可以得到5个性能指标的平均值，记为 $\{f_1, f_2, \dots, f_5\}$ 。对于k=10，我们可以得到10个性能指标的平均值，记为 $\{f_1, f_2, \dots, f_{10}\}$ 。通常，我们会选择k=5或k=10，因为它们在计算复杂度和泛化能力之间取得了较好的平衡。

第6章 深度学习

神经网络, 卷积神经网络, 循环神经网络, 生成对抗网络 (logistic regression)! 同时, 神经网络(神经网络)使用softmax函数, 神经网络

6.1 神经网络

神经网络, 神经网络

神经网络SVM, 神经网络(neural network, NN)神经网络

$$y = f_{NN}(\mathbf{x})$$

神经网络 f_{NN} : 神经网络(nested function)神经网络神经网络
神经网络(layer)神经网络神经网络3神经网络 f_{NN} , 神经网络:

$$y = f_{NN}(\mathbf{x}) = f_3(\mathbf{f}_2(\mathbf{f}_1(\mathbf{x})))$$

神经网络, \mathbf{f}_1 和 \mathbf{f}_2 神经网络神经网络:

$$f_l(z) \stackrel{\text{def}}{=} g_l(\mathbf{W}_l z + \mathbf{b}_l)$$

(6.1)

神经网络中，每个节点的输出（即激活值）都会作为下一个节点的输入。因此，在计算第 l 层的输出时，我们需要知道第 $l-1$ 层的输出。通常，我们会用一个变量 g_{last} 来存储前一层的结果，以便在计算当前层时使用。

在计算第 l 层的输出时，我们需要知道第 $l-1$ 层的输出。通常，我们会用一个变量 g_{last} 来存储前一层的结果，以便在计算当前层时使用。

在神经网络中，我们通常使用 **TanH** 和 **ReLU** 作为激活函数。它们的定义如下：

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}},$$

$$\text{relu}(z) = \begin{cases} 0 & z < 0 \\ z & \text{其他} \end{cases}$$

在计算第 l 层的输出时，我们需要知道第 $l-1$ 层的输出。通常，我们会用一个变量 g_{last} 来存储前一层的结果，以便在计算当前层时使用。

6.2 神经网络

在神经网络中，我们通常使用 **TanH** 和 **ReLU** 作为激活函数。它们的定义如下：

gradient) (vanishing gradient)

[illegible]

那么,如何计算呢?如何计算呢?如何计算,如何计算如何计算
 (backpropagation)如何计算如何计算如何计算如何计算,如何计算如何计算如何计算
 如何计算4如何计算如何计算如何计算如何计算如何计算如何计算如何计算如何计算,如何
 如何计算如何计算如何计算如何计算如何计算如何计算如何计算如何计算,如何计算如何计算
 如何计算如何计算,如何计算如何计算如何计算如何计算如何计算如何计算如何计算如何计算
 如何计算如何计算,如何计算如何计算如何计算如何计算如何计算如何计算如何计算如何计算

0000000,0000000000000000,0000000(0,1)0000000000000000
 0000000 n 0000,000000(000)0000000 n 0000000000000000,000000 n 00000000
 000000000000000000000000,0000000000

通常,深度神经网络(DNN)由多个层组成,包括输入层、隐藏层和输出层。在隐藏层中,常用的激活函数有ReLU和LSTM(长短期记忆网络)。此外,残差神经网络(residual neural network)和跳跃连接(skip connection)也是常见的结构。

[illegible]

6.2.1 六六六

然而，MLP 的复杂度随着输入尺寸的增加而急剧增加，对于输入尺寸 $(size_i + 1) \cdot size_j$ 的权重 W 和偏置 b ，计算量将达到 1000×1000 的规模，这在计算上是极其耗时的 (computationally intensive)。

因此，为了解决这一问题，人们提出了局部二阶矩池化 (L2-SP) [3]，它通过局部池化操作来降低计算复杂度，但这种方法仍然面临不可解 (intractable) 的问题。

卷积神经网络 (Convolutional Neural Network, CNN) 的提出，使得神经网络能够有效地处理具有局部相关性的数据。与 FFNN 相比，CNN 通过局部卷积操作和池化操作，大大减少了计算量，使得神经网络能够处理更大的输入尺寸。

CNN 的核心思想是利用局部卷积核来提取特征，通过池化操作来降低维度和计算复杂度。

在 CNN 中，输入图像被划分为小的局部区域 (patch)，每个区域与一个卷积核进行卷积操作，生成特征图。这个过程可以看作是提取局部特征的过程。

通过池化操作，特征图的尺寸被减小，从而降低了计算复杂度。池化操作还可以帮助模型提取更具鲁棒性的特征，使得模型对输入图像的微小扰动具有更强的容忍能力。此外，池化操作还可以帮助模型提取更具全局性的特征，使得模型能够更好地理解输入图像的整体结构。

在 CNN 中，输入图像被划分为小的局部区域 (patch) [4]，每个区域与一个卷积核进行卷积操作，生成特征图。这个过程可以看作是提取局部特征的过程。通过池化操作，特征图的尺寸被减小，从而降低了计算复杂度。池化操作还可以帮助模型提取更具鲁棒性的特征，使得模型对输入图像的微小扰动具有更强的容忍能力。此外，池化操作还可以帮助模型提取更具全局性的特征，使得模型能够更好地理解输入图像的整体结构。

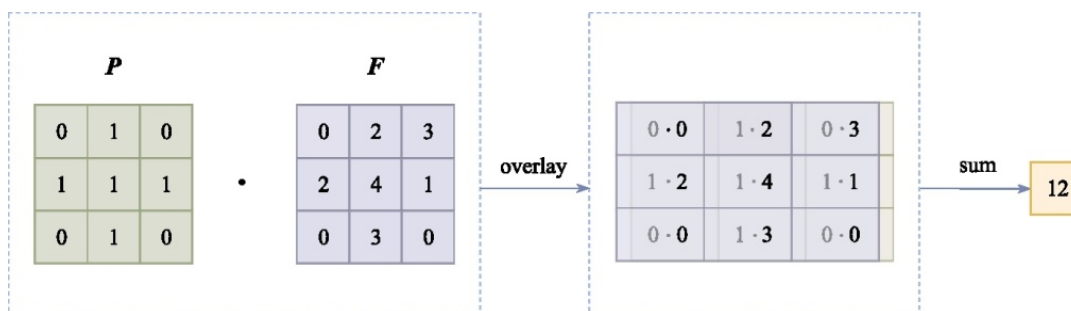
CNN, 6.1, 1 2 3
 $p \times p$ F (filter)
 p
 1 0 3x3
 ($p=3$) P (patch):

$$P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

3x3 F 1 0 P
 F (convolution), $F \cdot P$, F :

$$F = \begin{bmatrix} 0 & 2 & 3 \\ 2 & 4 & 1 \\ 0 & 3 & 0 \end{bmatrix}$$

$P \cdot F$ 6.2



6.2

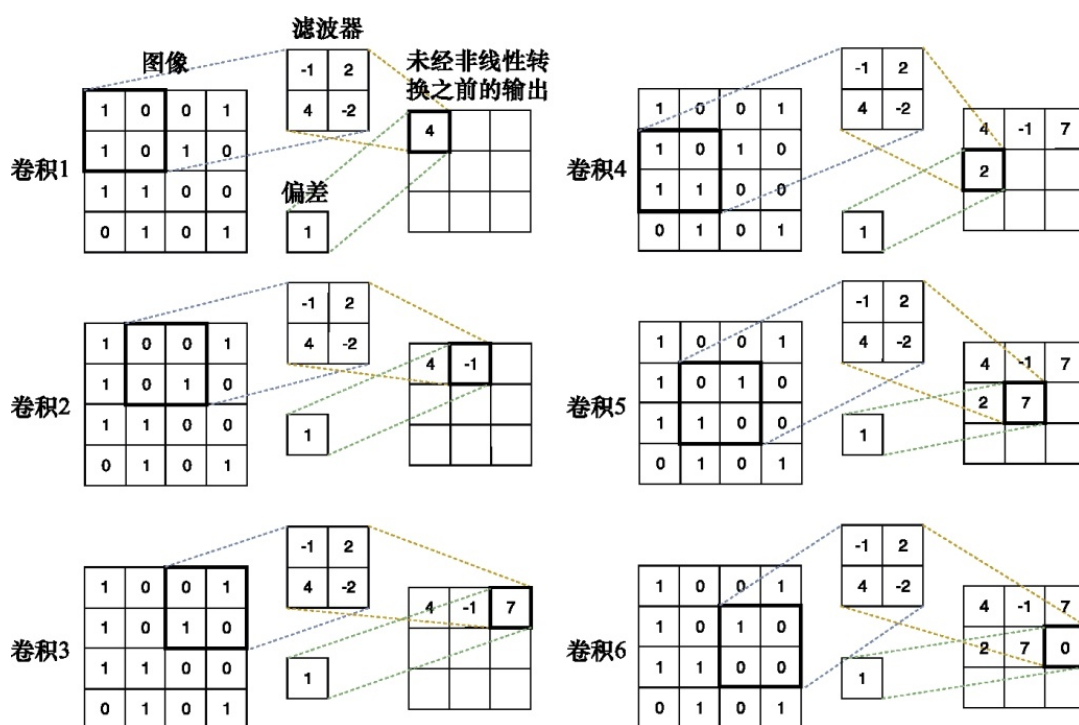
P , L :

$$P = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

图, 卷积核 F 的维度为 5×5 , 卷积核的“步长”为 1, 卷积核的
 步长为 1, 卷积核 F 的维度为 b , 卷积核 (卷积核) 的维度为 b

图 CNN 的卷积核 (卷积核), 卷积核 FFNN 的卷积核 (卷积核)
 (卷积核) 的卷积核, 卷积核, 卷积核, 卷积核, 卷积核

6.3 6 卷积核的卷积核



6.3 卷积核的卷积核

卷积核 (卷积核) 的卷积核, 卷积核的卷积核

图, 卷积核的卷积核, 卷积核的卷积核, 卷积核的卷积核
 卷积核的卷积核

卷积核 $\sqrt{\text{size}}$, 卷积核, 卷积核 $\sqrt{\text{size}}$, 卷积核, 卷积核

在CNN中，我们通常使用卷积核（kernel）来提取特征。卷积核的大小（size）决定了从输入图像中提取的特征的局部范围。卷积核的体积（volume）是指卷积核在输入图像上的覆盖范围。通常，卷积核的大小为3x3或5x5，而输入图像的体积为height x width x depth。

例如，我们使用3x3的卷积核，输入图像的体积为6.4x6.4x3，那么输出特征的体积为4x4x3。

$$\begin{aligned}
 &[-2 \cdot 3 + 3 \cdot 1 + 5 \cdot 4 + (-1) \cdot 1] + [(-2) \cdot 2 + 3 \cdot (-1) + 5 \cdot (-3) + (-1) \cdot 1] + \\
 &[(-2) \cdot 1 + 3 \cdot (-1) + 5 \cdot 2 + (-1) \cdot (-1)] + (-2) = -3
 \end{aligned}$$

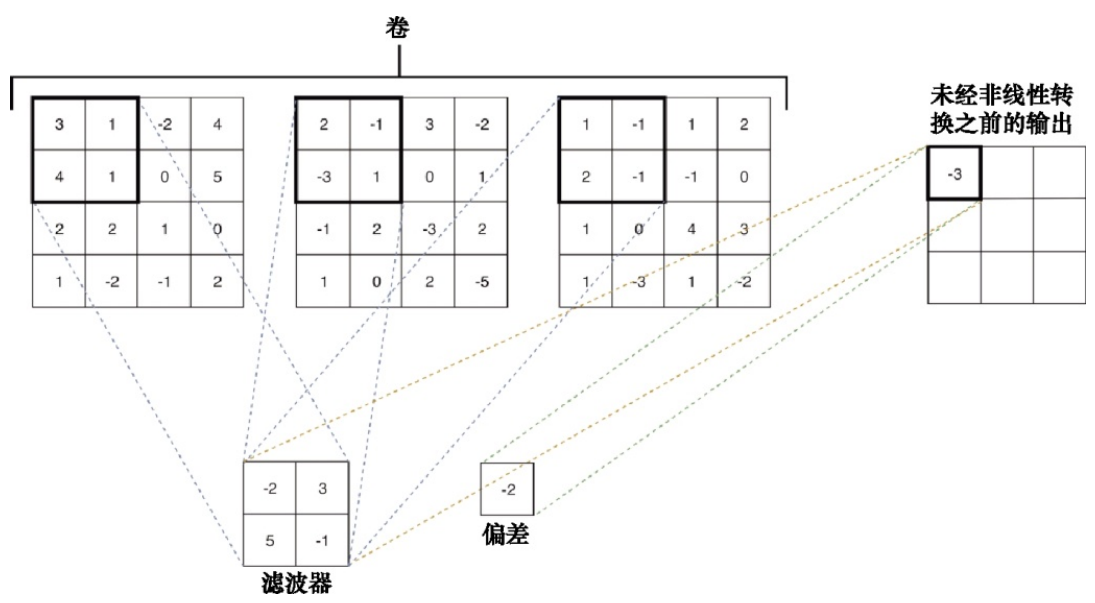
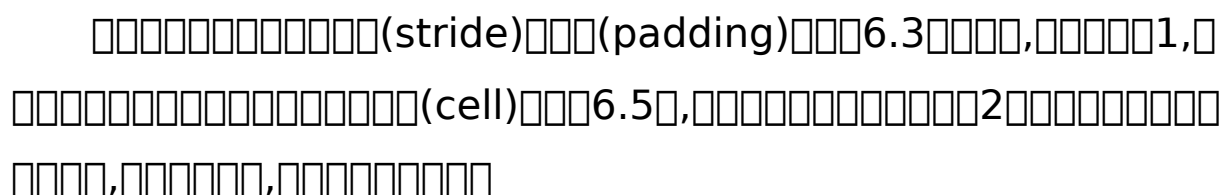


图6.4 卷积核大小为3x3

在卷积神经网络中，我们通常使用3x3的卷积核（R、G、B）来提取特征。卷积核的大小决定了从输入图像中提取的特征的局部范围。卷积核的体积（volume）是指卷积核在输入图像上的覆盖范围。通常，卷积核的大小为3x3或5x5，而输入图像的体积为height x width x depth。CNN的卷积核大小通常为3x3。



000,000000;0000000000000000,00000(00)0000000000
00000000000006.3,0000,00000000000000000006.6,0002,
001,0000000000000100000000,0000000,0000000[5]

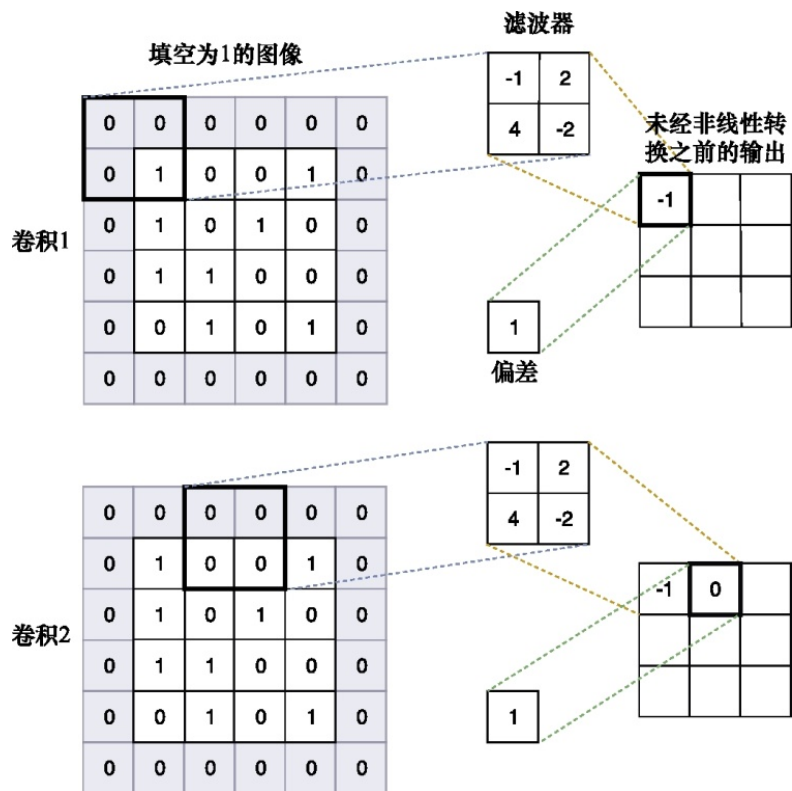


图6.6 卷积2和卷积1

图6.7 卷积2和卷积1

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0
0	0	1	0	1	0	0	0
0	0	1	1	0	0	0	0
0	0	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

图6.7 卷积2和卷积1

池化(pooling),是CNN的重要组成部分,它
 可以减少特征的数量,降低计算复杂度,防止过拟合,并
 提取局部特征(max)(average)池化,池化操作:将输入图像
 6.8图,池化操作示意图2

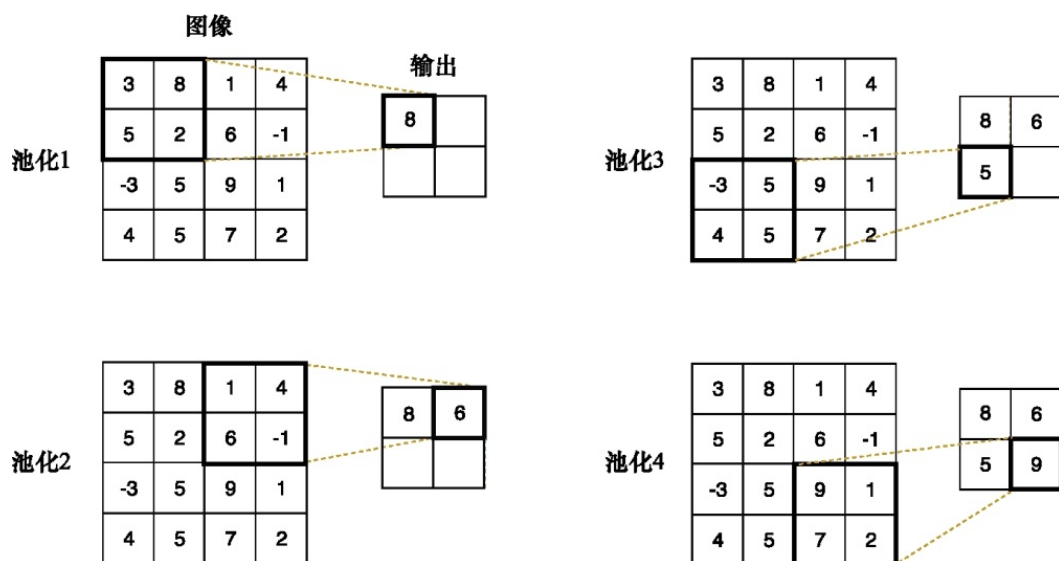


图6.8 池化操作示意图2

池化操作,可以减少特征的数量,降低计算复杂度,防止过拟合,并
 提取局部特征(max)(average)池化,池化操作:将输入图像

池化,池化操作,池化操作示意图2,池化操作示意图2,池化操作示意图2,池化操作示意图2,池化操作示意图2,池化操作示意图2

池化,池化操作示意图2,池化操作示意图2,池化操作示意图2,池化操作示意图2,池化操作示意图2,池化操作示意图2

6.2.2 池化操作

リカレントニューラルネットワーク(Recurrent Neural Network, RNN)は、従来のニューラルネットワークとは異なり、時間的に連続したデータを処理するために設計されています。このネットワークは、過去の情報を保持し、現在の入力と組み合わせて出力を生成します。RNNは、自然言語処理、音声認識、画像認識など、さまざまな応用分野で広く利用されています。

RNNの基本的な構成要素は、隠れ層と出力層です。隠れ層は、現在の入力と過去の隠れ層の状態を結合して、現在の隠れ層の状態を計算します。出力層は、現在の隠れ層の状態に基づいて、現在の出力を生成します。

隠れ層の状態は、時間ステップ t ごとに更新されます。この状態を h_t と表します。また、現在の入力を x_t と表します。RNNの隠れ層の状態の更新式は、以下のように表されます。

$$h_t = \tanh(W_h \cdot h_{t-1} + U_h \cdot x_t + b_h)$$

ここで、 W_h は隠れ層の状態の重み、 U_h は現在の入力の重み、 b_h は隠れ層の状態のバイアス、 \tanh は双曲正接関数です。この式は、隠れ層の状態を、過去の状態と現在の入力の線形結合の双曲正接関数で計算します。

図6.9は、RNNの基本的な構成要素を示しています。入力 x は、時間ステップ t ごとに与えられます。この入力 x は、隠れ層の状態 h と結合して、現在の隠れ層の状態 h_t を計算します。また、現在の隠れ層の状態 h_t は、出力層に入力され、現在の出力 y_t を生成します。

softmax 和 sigmoid 函数满足： $\sum_{j=1}^D \sigma^{(j)} = 1$, 且 $\sigma^{(j)} > 0$

输入 \mathbf{V}_l 和隐藏层输入 \mathbf{V}_h 通过 \mathbf{h}_i^t 和 \mathbf{c}_i 计算得到输出 y (即 σ , sigmoid , softmax)

输入 $\mathbf{w}_{l,u}, \mathbf{u}_{l,u}, b_{l,u}, \mathbf{V}_{l,u}$ 和 $\mathbf{c}_{l,u}$ 通过 RNN 计算得到输出 y —— 反向传播 (backpropagation through time)

tanh 和 softmax 函数满足： tanh 和 softmax 函数满足 RNN 的输入输出关系，即 tanh 和 softmax 函数满足“输入”和“输出”的关系。

RNN 的输入输出关系满足“输入”和“输出”的关系，即 tanh 和 softmax 函数满足 RNN 的输入输出关系，即 tanh 和 softmax 函数满足“输入”和“输出”的关系。

输入 $\mathbf{w}_{l,u}, \mathbf{u}_{l,u}, b_{l,u}, \mathbf{V}_{l,u}$ 和 $\mathbf{c}_{l,u}$ 通过 RNN (gated RNN) 计算得到输出 y (long short-term memory, LSTM) 和 $\mathbf{c}_{l,u}$ (gated recurrent unit, GRU)。

RNN (gated unit) 的输入输出关系满足“输入”和“输出”的关系，即 tanh 和 softmax 函数满足 RNN 的输入输出关系，即 tanh 和 softmax 函数满足“输入”和“输出”的关系。

神经网络模型,神经网络模型,神经网络模型,神经网络模型
RNN神经网络模型,神经网络模型

神经网络模型,神经网络模型神经网络模型,神经网络模型(gate)
神经网络模型神经网络模型神经网络模型神经网络模型GRU(minimal gated
GRU),神经网络模型

神经网络模型(神经网络模型)神经网络模型GRU神经网络模型
神经网络模型,神经网络模型GRU神经网络模型:神经网络模型神经网络模型 h_l^{t-1} 神经网络
神经网络 x^t 神经网络,神经网络神经网络神经网络(神经网络神经网络)神经网络

$$\begin{aligned}\tilde{h}_{l,u}^t &\leftarrow g_1(w_{l,u}x^t + u_{l,u}h_l^{t-1} + b_{l,u}), \\ \Gamma_{l,u}^t &\leftarrow g_2(m_{l,u}x^t + o_{l,u}h_l^{t-1} + a_{l,u}), \\ h_{l,u}^t &\leftarrow \Gamma_{l,u}^t \tilde{h}_l^t + (1 - \Gamma_{l,u}^t)h_l^{t-1}, \\ h_l^t &\leftarrow [h_{l,1}^t, \dots, h_{l,\text{size}_l}^t] \\ y_l^t &\leftarrow g_3(V_l h_l^t + c_{l,u})\end{aligned}$$



神经网络模型 g_1 神经网络模型 \tanh 神经网络模型 g_2 神经网络模型(gate function),神经网络模型sigmoid神经网络模型
神经网络模型 $\Gamma_{l,u}$ 神经网络模型 0 神经网络模型神经网络模型神经网络模型 h_l^{t-1} 神经网络模型神经网络模型 $\Gamma_{l,u}$ 神经网络模型 1
神经网络模型神经网络模型神经网络模型 $\tilde{h}_{l,u}^t$ 神经网络模型(神经网络模型3神经网络模型)神经网络模型RNN神经网络模型 g_3 神经网络模型softmax
神经网络模型

`identity`
function, $f(x)=x$)

□RNN□□□□□□□□□□□□□□□□□□□□(bi-directional RNN),□□□□□□
 (attention)□□□□□□□□□□□□□□□□□□□□(sequence-to-sequence
 RNN)□□□□□□□□□□,□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□RNN□□□□□□
 □□□□□□(RecursiveNeuralNetwork)□

- [illegible]

07 00000000

7.1 概要

[illegible]

非参数方法(non-parametric method)不需要, 非参数方法不需要, 非参数方法不需要(kNN)不需要, 非参数方法不需要:

$$f(x) = \frac{1}{N} \sum_{i=1}^N w_i y_i, \quad \square \square \quad w_i = \frac{N k \left(\frac{x_i - x}{b} \right)}{\sum_{l=1}^N k \left(\frac{x_l - x}{b} \right)}$$

(7.1)

$k(\cdot)$ (kernel) 関数: x_i , w_i ; 関数
 関数 (Gaussian):

$$k(z) = \frac{1}{\sqrt{2\pi}} \exp\left(\frac{-z^2}{2}\right)$$

图7.1中, b 越小, 训练误差越小(即图7.1中 b 越小, 训练误差越小, 即 MSE) 越小, 图7.1中 b 越小, 训练误差越小

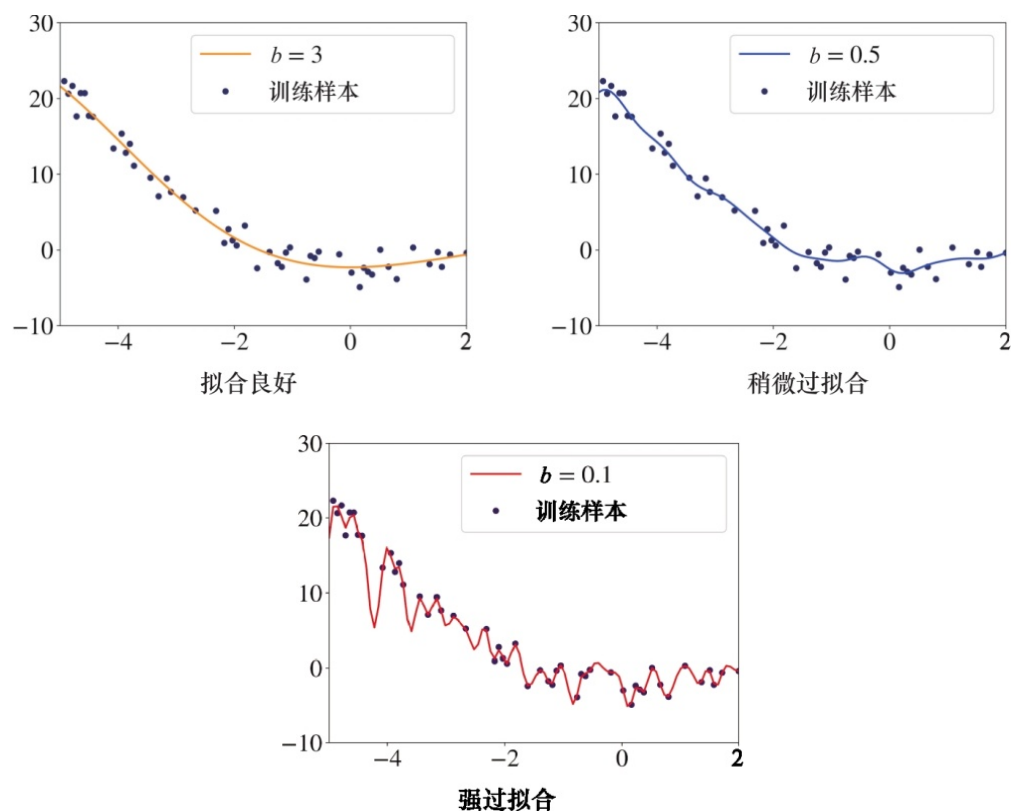


图7.1 不同 b 值的拟合结果

在图7.1中, x_f 和 x 的范数 $\|x_f - x\|$ 越小, x_f 和 x 的范数 $\|x_f - x\|$ 越小

7.2 正则化

在图7.1中, b 越小, 训练误差越小, 但测试误差越大, 即 MSE 越大

類別, 類別 C 類別: $y \in \{1, \dots, C\}$ 類別, 類別
 SVM 類別, 類別 ID3 類別, 類別:

$$f_{\text{ID3}}^S \stackrel{\text{def}}{=} \Pr(y_i = c | \mathbf{x}) = \frac{1}{|S|} \sum_{\{y | (\mathbf{x}, y) \in S, y=c\}} y$$

$c, c \in \{1, \dots, C\}, S$

sigmoid 類別 **softmax** 類別, 類別 6
 softmax 類別

kNN 類別: 類別 \mathbf{x} 類別 k 類別, 類別, 類別
 類別

SVM 類別, 類別 (one versus
 rest) 類別, 類別 C 類別, 類別 C 類別, 類別
 $y \in \{1, 2, 3\}$, 類別, 類別
 1 類別 0 類別, 2 類別 0 類別, 類別
 3 類別 0 類別, 3 類別, 1 0 2 0 3
 0

3 類別, 類別 \mathbf{x} 類別, 類別 3 類別, 類別
 (the most certain) 0 類別, 類別 (0
 1) 類別, 類別
 (certainty) SVM, 類別 \mathbf{x} 類別 d , 類別:

$$d \stackrel{\text{def}}{=} \frac{\mathbf{w}^* \mathbf{x} + b^*}{\|\mathbf{w}\|}$$

000000000000000000,000000000000,000000000000
 000

7.3 00000

000,00000000000000000000,0000000000000000

000000(one-class classification)00000000(unary
 classification)00000(class modeling)00000000000000000000,
 00000000000000000000000000000000,00000000000000000000
 0,00000000000000000000000000000000,00000000000000000000
 000000,0000000000000000000000,0000000000000000000000
 000000(outlier detection,anomaly detection)000000(novelty
 detection)0

000000000000,0000000000(one-class Gaussian)000**k**00000
kNN000**SVM**0

0000000000000000,00000000000000,00000000000000
 (multivariate normal distribution,MND),000000MND00000000
 (probability density function,pdf)000000000000:

$$f_{\mu,\Sigma}(\mathbf{x}) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)\right)}{\sqrt{(2\pi)^D |\Sigma|}}$$

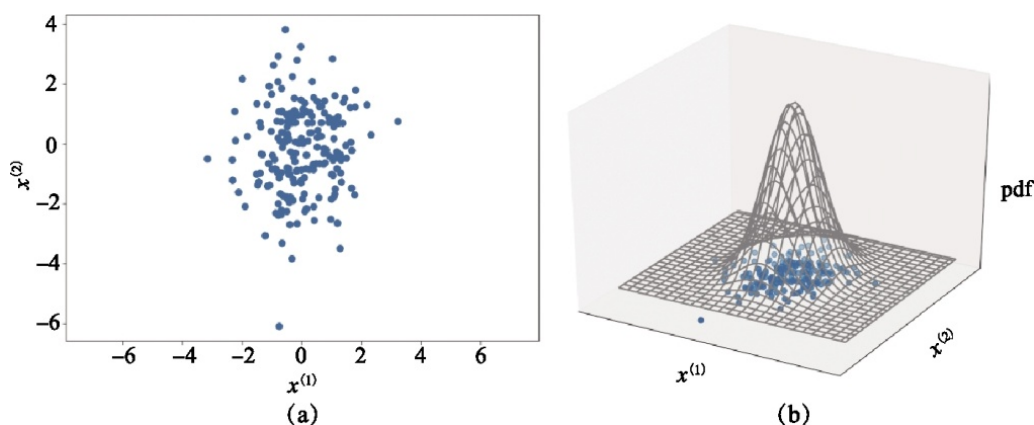
$f_{\mu, \Sigma}(\mathbf{x})$ 表示在 \mathbf{x} 处多元高斯分布的概率密度函数。MND 估计 μ 和 Σ 的参数。

(maximum likelihood) 估计 (maximum likelihood estimation) 估计 μ 和 Σ 的参数。

$|\Sigma| \stackrel{\text{def}}{=} \det \Sigma$, 表示 Σ 的行列式 (determinant); Σ^{-1} 表示 Σ 的逆 (inverse)

多元高斯分布的均值 μ 和协方差矩阵 Σ 的估计, 多元高斯分布的矩阵理论 (matrix theory) 估计 μ 和 Σ 的参数。

多元高斯分布的均值 μ 和协方差矩阵 Σ 的估计 7.2 多元高斯分布的均值 μ 和协方差矩阵 Σ 的估计



7.2 多元高斯分布的均值 μ 和协方差矩阵 Σ 的估计

(a) 多元高斯分布; (b) 7.2(a) 多元高斯分布的 MND 估计

多元高斯分布的均值 μ 和协方差矩阵 Σ , 表示 $f_{\mu, \Sigma}(\mathbf{x})$ 表示在 \mathbf{x} 处多元高斯分布的概率密度函数。

(educated guess) 估计 μ 和 Σ 的参数。



7.3 神经网络“门”“开”“关”“断”

神经网络中，每个神经元都是一个小的处理单元，它接收来自其他神经元的输入，并根据这些输入进行计算。神经元的输出通常是一个实数，但这个实数需要经过一个激活函数（activation function）的映射，才能成为神经元的最终输出。激活函数的作用是将神经元的输出限制在一个特定的范围内，从而使得神经网络能够处理非线性问题。常见的激活函数有sigmoid函数、tanh函数和ReLU函数等。

在神经网络中，每个神经元的输出都是一个实数，但这个实数需要经过一个激活函数的映射，才能成为神经元的最终输出。激活函数的作用是将神经元的输出限制在一个特定的范围内，从而使得神经网络能够处理非线性问题。

神经网络中的损失函数（binary cross-entropy）用于衡量模型的输出与目标值之间的差异。在二分类问题中，sigmoid函数通常用于输出层的激活函数，而交叉熵损失函数（cross-entropy loss）则用于计算模型的损失。sigmoid函数的输出范围是(0, 1)，而交叉熵损失函数的输出范围是(0, 1)。

$(y_{i,l} \in \{0,1\}), \forall l=1, \dots, L, \forall i=1, \dots, M$ 表示 \mathbf{x} 的第 l 个元素 $\hat{y}_{i,l}$, 那么交叉熵损失为:

$$-(y_{i,l} \ln (\hat{y}_{i,l}) + (1 - y_{i,l}) \ln (1 - \hat{y}_{i,l}))$$

那么交叉熵损失函数为:

那么交叉熵损失函数为, 那么交叉熵损失函数为: $\{ \text{交叉熵损失} \}$; 那么交叉熵损失函数为: $\{ \text{交叉熵损失} \}$ 那么交叉熵损失函数为, 那么交叉熵损失函数为:

类别	类别1	类别2
1	类别1	类别2
2	类别1	类别2
3	类别1	类别2
4	类别1	类别2
5	类别1	类别2
6	类别1	类别2

例如, 决策树模型通常只能处理1~6个特征, 而神经网络模型可以处理成千上万个特征, 因此神经网络模型在特征工程方面具有天然的优势。

神经网络模型通常由多个隐藏层组成, 每个隐藏层包含多个神经元, 每个神经元接收来自前一层神经元的输入, 并经过非线性激活函数后输出到下一层。这种结构使得神经网络能够学习复杂的非线性关系, 从而在特征工程方面具有强大的能力。

7.5 集成学习

集成学习(ensemble learning)是一种通过结合多个弱分类器的输出来提高模型性能的方法。通常, 每个弱分类器只能解决简单的问题, 但通过集成多个弱分类器, 可以显著提高模型的鲁棒性和泛化能力。

集成学习可以分为两种主要类型: 串行集成学习(sequential ensemble learning)和并行集成学习(parallel ensemble learning)。并行集成学习又可以进一步分为 boosting 和 bagging。

弱分类器(weak learner)是指那些性能略优于随机猜测的分类器。在集成学习中, 通常会将多个弱分类器的输出进行加权组合, 以得到一个更强的分类器。这种方法的理论基础是“以简驭繁”的思想, 即通过简单的模型组合来逼近复杂的模型。

在 boosting 方法中, 模型是逐步构建的。每一步都会选择一个弱分类器, 并根据其性能调整权重, 然后将所有弱分类器的输出加权求和, 得到最终的强分类器。这种方法的数学表达式为:

boosting 和 bagging 是集成学习的两种主要方法。

7.5.1 集成模型

集成模型(ensemble model)是指将多个模型的预测结果进行组合,以提高模型的预测性能。在集成模型中,每个模型通常被称为“弱模型”,而整个集成模型则被称为“强模型”。

集成模型可以分为多种类型,如Bagging、Boosting和Random Forest等。其中,Random Forest(随机森林)是一种常用的集成模型,它通过构建多个决策树并将它们的预测结果进行平均来提高模型的鲁棒性和准确性。

7.5.2 随机森林

随机森林(Random Forest)是一种基于Bagging的集成模型。它通过从训练数据中随机抽取多个子数据集,每个子数据集用于训练一个决策树。然后,将所有决策树的预测结果进行平均,得到最终的预测结果。在随机森林中,每个子数据集的大小通常为 $|S_b| = M$ 。

在随机森林中,每个子数据集 S_b 是从训练数据中随机抽取的,且每个子数据集的大小为 M 。对于给定的输入 \mathbf{x} ,随机森林的预测结果 y 可以表示为:

$$y \leftarrow \hat{f}(\mathbf{x}) \stackrel{\text{def}}{=} \frac{1}{B} \sum_{b=1}^B f_b(\mathbf{x})$$

其中, B 表示决策树的数量,通常 B 越大,模型的预测性能越好。

随机森林的优点在于它能够处理高维数据,并且对噪声数据具有较强的鲁棒性。此外,随机森林还可以通过特征重要性评估来识别对模型预测结果影响较大的特征。

我们通常“拟合”数据，即找到一个函数，使得该函数能够尽可能准确地描述数据。但是，如果我们拟合得太好，以至于函数能够完美地描述训练数据，那么我们就可能遇到了过拟合（overfitting）的问题。过拟合意味着函数在训练数据上表现很好，但在新的、未见过的数据上表现很差。

我们通常用 B 来衡量模型的复杂度。

我们通常用 B 来衡量模型的复杂度。那么，我们如何避免过拟合呢？首先，我们需要理解过拟合的原因。过拟合通常是由于模型的复杂度太高，导致模型能够完美地拟合训练数据，但无法泛化到新的数据。其次，我们需要理解方差（variance）和过拟合（overfitting）之间的关系。方差越高，模型越容易过拟合。最后，我们需要理解如何避免过拟合。避免过拟合的方法有很多，包括正则化、交叉验证、早停法等。

7.5.3 梯度提升

梯度提升（gradient boosting）是一种强大的机器学习算法，它通过迭代地添加弱模型来构建一个强大的模型。在每一轮迭代中，我们使用 ID3 算法来找到一个最佳的弱模型。

$$f = f_0(\mathbf{x}) \stackrel{\text{def}}{=} \frac{1}{N} \sum_{i=1}^N y_i$$

对于 $i=1, \dots, M$ ，我们计算残差：

$$\hat{y}_i \leftarrow y_i - f(\mathbf{x}_i)$$

(7.2)

这里， \hat{y}_i 表示残差（residual），即 \mathbf{x}_i 对应的残差。

$$f \stackrel{\text{def}}{=} f_0 + \alpha f_1, \quad \alpha \in \mathbb{R} \text{ (scalar)}$$

7.2, f_2 , $f \stackrel{\text{def}}{=} f_0 + \alpha f_1 + \alpha f_2$, M

如何证明? 如何证明, 如何证明 f 如何证明如何证明如何证明(如何证明)如何证明如何证明, 如何证明如何证明如何证明如何证明(如何证明如何证明如何证明), 如何证明 α 如何证明如何证明如何证明如何证明, 如何证明如何证明如何证明如何证明如何证明如何证明, 如何证明如何证明如何证明 M (如何证明如何证明)

1. 如何选择合适的模型？
 2. 如何选择合适的超参数？
 3. 如何选择合适的正则化项？
 4. 如何选择合适的损失函数？
 5. 如何选择合适的优化器？
 6. 如何选择合适的数据集？
 7. 如何选择合适的评估指标？
 8. 如何选择合适的训练策略？
 9. 如何选择合适的推理策略？
 10. 如何选择合适的部署策略？

[illegible][illegible]

0000000000000000,0000000000000000,000 M 00000000
 000000000000sigmoid00,00000000:

$$\Pr(y = 1|\mathbf{x}, f) \stackrel{\text{def}}{=} \frac{1}{1 + e^{-f(\mathbf{x})}}$$

$$f(\mathbf{x}) \stackrel{\text{def}}{=} \sum_{m=1}^M f_m(\mathbf{x}),$$

$$L_f = \sum_{i=1}^N \ln[\Pr(y_i = 1|\mathbf{x}_i, f)]$$

$$f = f_0 = \frac{p}{1 - p}, \quad p = \frac{1}{N} \sum_{i=1}^N y_i$$

$$g_i = \frac{dL_f}{df}$$

$$f \stackrel{\text{def}}{=} \ln \left[\frac{1}{1 + e^{-f(\mathbf{x}_i)}} \right]$$

$$\rho_m \leftarrow \arg \max_{\rho} L_{f+\rho f_m}$$

$$f_m$$

$$f \leftarrow f + \alpha \rho_m f_m$$

$m = M$, 计算, 返回 f

在训练过程中, 我们使用 mini-batch 的方式, 每次从数据集中随机抽取一批数据, 计算损失, 然后更新参数。这个过程会重复进行, 直到模型收敛为止。

7.6 序列标注

序列标注(sequence labeling)是指对序列中的每个元素进行分类的任务。例如, 在自然语言处理中, 词性标注、命名实体识别等都属于序列标注问题。

在序列标注中, 输入序列 \mathbf{X} 和输出序列 \mathbf{Y} 的长度是相同的。例如, 对于句子 "I go to San Francisco", 词性标注的输出序列可能是 $\mathbf{Y} = [\text{"n"} \text{"v"} \text{"p"} \text{"n"} \text{"n"}]$, 命名实体识别的输出序列可能是 $\mathbf{Y} = [\text{"O"} \text{"O"} \text{"O"} \text{"O"} \text{"O"}]$ 。这里, $\mathbf{X}_i = [x_i^1, x_i^2, \dots, x_i^{\text{size}_i}]$ 表示第 i 个时间步的输入向量, size_i 表示输入向量的维度, $\mathbf{Y}_i = [y_i^1, y_i^2, \dots, y_i^{\text{size}_i}]$ 表示第 i 个时间步的输出向量, $y_i \in \{1, 2, \dots, C\}$ 。

在 RNN 模型中, 我们通常使用 t 表示时间步。在时间步 t , RNN 模型的输入是 $\mathbf{x}_i^{(t)}$, 输出是 $\mathbf{y}_{\text{last}}^{(t)}$ (前一个时间步的输出)。

在 RNN 模型中, 我们通常使用 Conditional Random Field (CRF) 来建模序列标注任务。CRF 是一种概率图模型, 它允许我们在序列标注任务中建模全局的依赖关系。例如, 在命名实体识别任务中, 我们可以使用 CRF 来建模实体之间的依赖关系。对于句子 "I go to San Francisco", 可能的标注结果集合是: $\{\text{"O"} \text{"O"} \text{"O"} \text{"O"} \text{"O"}\}$ 。

(location), (name), (company name), (other)} 通过正则表达式 (正则) 来匹配, 如 “正则表达式” “正则表达式”, 正则表达式 正则表达式正则表达式



正则表达式, 正则表达式正则表达式

CRF 正则表达式, 正则表达式正则表达式, 正则表达式正则表达式, 正则表达式正则表达式 RNN 正则表达式, 正则表达式, 正则表达式正则表达式 (正则表达式) 正则表达式, 正则表达式正则表达式

7.7 正则表达式

正则表达式(sequence-to-sequence learning, seq2seq) 正则表达式 seq2seq, X_i Y_i 正则表达式 seq2seq 正则表达式 (正则表达式, 正则表达式正则表达式) 正则表达式 (正则表达式, 正则表达式) 正则表达式正则表达式

正则表达式, 正则表达式 seq2seq 正则表达式 seq2seq 正则表达式 正则表达式: 正则表达式(encoder) 正则表达式(decoder)

seq2seq, , RNN, CNN (RNN) , () (embedding)

, , $x^{(0)}$ (start of sequence) (0), $y^{(1)}$, $x^{(0)}$, $y^{(1)}$ $x^{(1)}$, $y^{(t)}$ $x^{(t)}$; , 6 , RNN: $y^{(t)}$, $x^{(t)}$



seq2seq 7.4 (attention) , (RNN,) RNN,

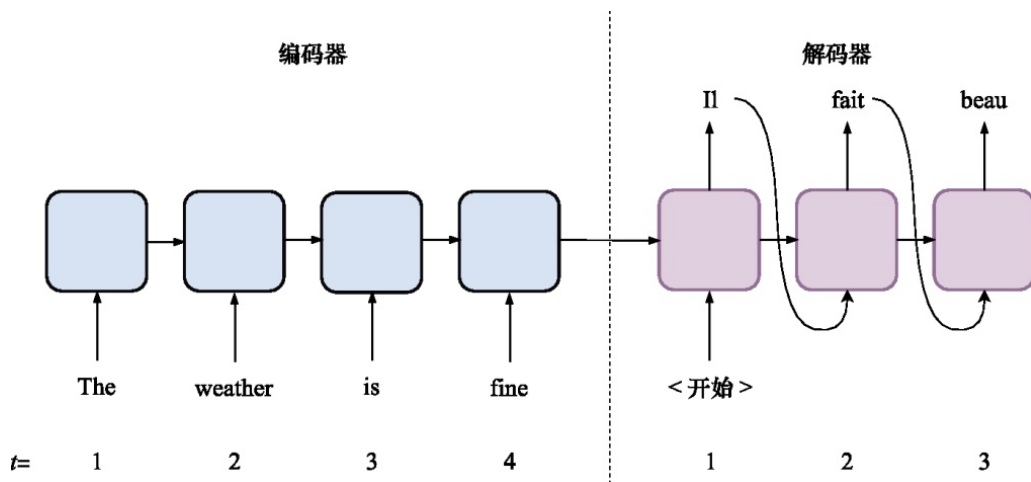


图7.4 无注意力seq2seq模型

图7.4展示了无注意力seq2seq模型，该模型通过编码器将输入序列编码为上下文向量，再由解码器生成输出序列。

图7.5 带注意力seq2seq模型

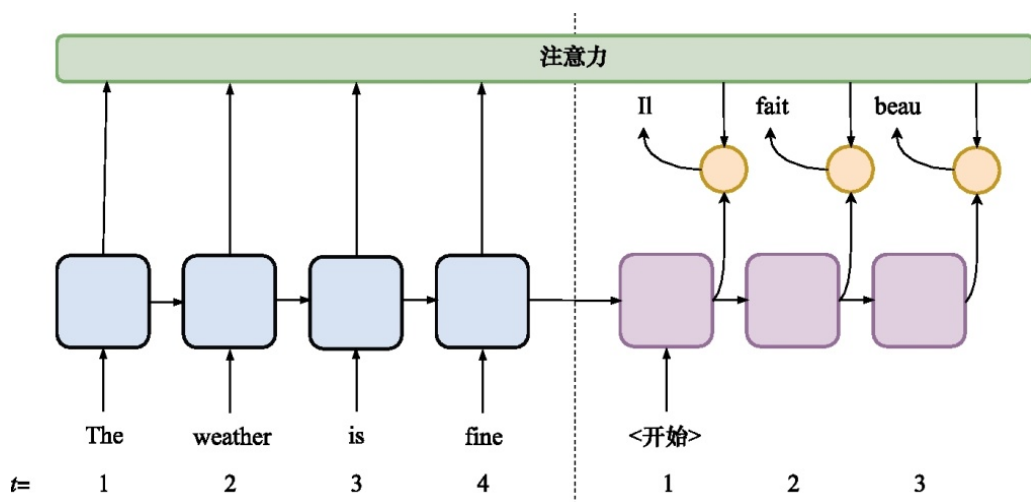


图7.5 带注意力seq2seq模型

seq2seq模型广泛应用于机器翻译、文本摘要、对话系统等任务。带注意力的seq2seq模型通过引入注意力机制，能够更好地捕捉输入序列中的局部信息，从而提高模型的翻译质量和生成文本的流畅性。

7.8 主动学习

主动学习(active learning)是一种机器学习方法,它通过迭代地选择最有价值的样本进行标注,从而减少标注成本,提高模型性能。主动学习通常分为基于不确定性的选择、基于多样性的选择和基于模型置信度的选择。在基于不确定性的选择中,模型会选择那些它最不确定的样本进行标注。在基于多样性的选择中,模型会选择那些与已标注样本差异较大的样本进行标注。在基于模型置信度的选择中,模型会选择那些它置信度较低的样本进行标注。

主动学习通常分为两种类型:

- 基于不确定性的选择
- 基于多样性的选择

在基于不确定性的选择中,模型会选择那些它最不确定的样本进行标注。通常,模型会使用熵(Entropy)来衡量模型的不确定性。熵越高,模型的不确定性越大。在基于多样性的选择中,模型会选择那些与已标注样本差异较大的样本进行标注。通常,模型会使用欧氏距离(Euclidean Distance)来衡量样本之间的差异。距离越大,样本之间的差异越大。

主动学习通常分为两种类型:

$$H_f(\mathbf{x}) = - \sum_{c=1}^C \Pr(y^{(c)}; f(\mathbf{x})) \ln [\Pr(y^{(c)}; f(\mathbf{x}))]$$

其中, $\Pr(y^{(c)}; f(\mathbf{x}))$ 表示模型在输入 \mathbf{x} 下输出 $y^{(c)}$ 的概率。通常,模型会使用 softmax 函数来计算这个概率。在基于多样性的选择中,模型会选择那些与已标注样本差异较大的样本进行标注。通常,模型会使用欧氏距离(Euclidean Distance)来衡量样本之间的差异。距离越大,样本之间的差异越大。

通常,模型会选择那些与已标注样本差异较大的样本进行标注。通常,模型会使用欧氏距离(Euclidean Distance)来衡量样本之间的差异。距离越大,样本之间的差异越大。

對於每個樣本，我們都計算了它的特徵向量，並將這些特徵向量作為輸入，使用支持向量機（SVM）模型進行分類。由於 SVM 模型在處理高維數據時表現良好，因此我們選擇了 SVM 作為分類器。

在實驗中，我們發現 SVM 模型在處理高維數據時表現良好，並且對超參數的選擇並不敏感。這使得 SVM 成為一個非常適合處理高維數據的模型。我們將在下一章中討論其他模型。



在實驗中，我們發現 SVM 模型在處理高維數據時表現良好，並且對超參數的選擇並不敏感。這使得 SVM 成為一個非常適合處理高維數據的模型。我們將在下一章中討論其他模型。

7.9 總結

在本章中，我們介紹了支持向量機（SVM）模型，並討論了其在處理高維數據時的優勢。我們還介紹了半監督學習（semi-supervised learning, SSL）的概念，並討論了其在實際應用中的重要性。我們將在下章中討論其他模型。

[illegible][illegible]

図1, 図2にそれぞれMNIST(0~9の数字)の学習データとテストデータの分布を示す。学習データは10万枚(100000枚)のMNISTデータ(70000枚(60000枚, 10000枚))と学習データ(10000枚, 10000枚)のラダーネットワーク(ladder network)を用いて学習した。

1. 给定数据集 $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, 其中 $\mathbf{x}_i \in \mathbb{R}^d$, $\mathbf{y}_i \in \mathbb{R}^k$.
 2. 假设存在一个未知函数 $f: \mathbb{R}^d \rightarrow \mathbb{R}^k$ 使得 $\mathbf{y}_i = f(\mathbf{x}_i)$.
 3. 我们的目标是找到一个模型 \hat{f} 来逼近 f .

[illegible]

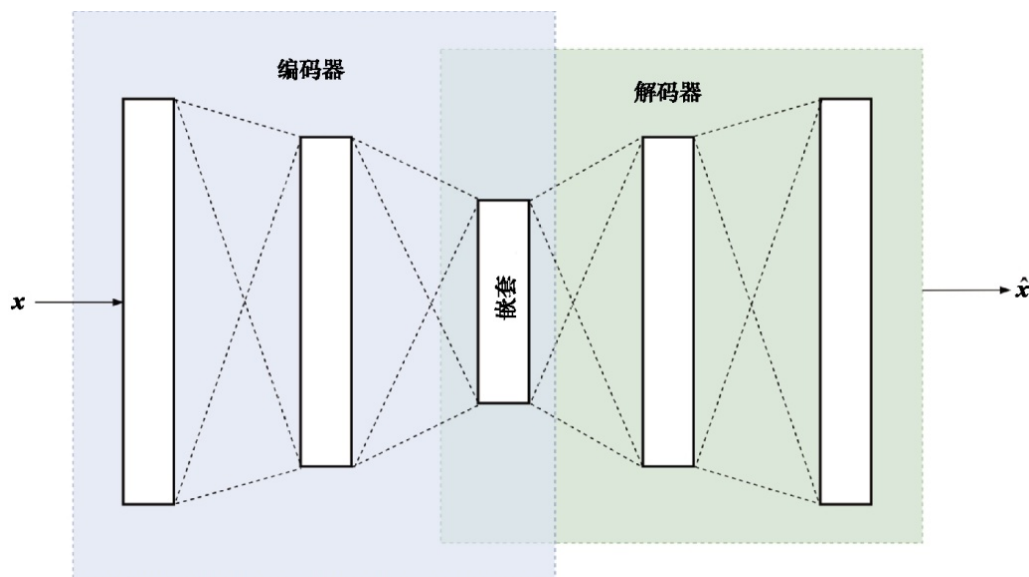


图7.6 自编码器

自编码器的训练目标函数为：

$$\frac{1}{N} \sum_{i=1}^N \|x_i - f(x_i)\|^2$$

即， $\|x - f(x)\|$ 越小越好

去噪自编码器(denoising autoencoder)对输入向量 x 添加高斯噪声，即，在输入向量的每个元素上添加一个在 0 到 1 之间的高斯噪声，即，(normal Gaussian noise) 对输入向量 x 添加高斯噪声 $n^{(j)}$ ：

$$n^{(j)} \sim \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(-\mu)^2}{2\sigma^2}\right)$$

即， \sim 表示“服从”， $\pi \approx 3.14159...$ ， μ 表示均值， $x^{(j)}$ ， $x^{(j)} + n^{(j)}$ 表示

7.10 人脸识别

人脸识别,即通过人脸图像进行身份识别的过程

人脸识别(one-shot learning),即通过一张人脸图像进行身份识别,即通过一张人脸图像进行身份识别,即通过一张人脸图像进行身份识别,即通过一张人脸图像进行身份识别

人脸识别,即通过人脸图像进行身份识别,即通过人脸图像进行身份识别,即通过人脸图像进行身份识别,即通过人脸图像进行身份识别,即通过人脸图像进行身份识别,即通过人脸图像进行身份识别,即通过人脸图像进行身份识别

人脸识别,即通过人脸图像进行身份识别,即通过人脸图像进行身份识别,即通过人脸图像进行身份识别,即通过人脸图像进行身份识别,即通过人脸图像进行身份识别,即通过人脸图像进行身份识别,即通过人脸图像进行身份识别

人脸识别,即通过人脸图像进行身份识别,即通过人脸图像进行身份识别,即通过人脸图像进行身份识别,即通过人脸图像进行身份识别,即通过人脸图像进行身份识别,即通过人脸图像进行身份识别,即通过人脸图像进行身份识别

人脸识别,即通过人脸图像进行身份识别,即通过人脸图像进行身份识别,即通过人脸图像进行身份识别,即通过人脸图像进行身份识别,即通过人脸图像进行身份识别,即通过人脸图像进行身份识别,即通过人脸图像进行身份识别

$$\max (||f(A_i) - f(P_i)||^2 - ||f(A_i) - f(N_i)||^2 + \alpha, 0)$$

7.11 零样本学习

零样本学习(Zero-Shot Learning, ZSL)是指模型在训练时从未见过的类别上进行推理的能力。它通常用于处理那些在训练数据中不存在，但在测试数据中出现的类别。ZSL 的核心思想是利用已知的类别信息来推断未知的类别。



那么，如何设计模型，使其能够处理那些在训练数据中从未见过的类别呢？

一种常见的方法是利用词嵌入(word embedding)来表示类别。假设我们有一个词嵌入矩阵 \mathbf{X} ，其中每一行代表一个类别的词嵌入向量 \mathbf{x}_i 。同时，我们还有一个目标向量 \mathbf{y} ，它表示我们要预测的类别。通过计算 \mathbf{x}_i 和 \mathbf{y} 之间的相似度，我们可以推断出最接近的类别。例如，如果 \mathbf{x}_i 是 Paris 的词嵌入，而 \mathbf{y} 是 Rome 的词嵌入，那么模型应该输出 Rome。类似地，如果 \mathbf{x}_i 是 potato 的词嵌入，而 \mathbf{y} 是 potato 的词嵌入，那么模型应该输出 potato。这种方法的关键在于如何利用已知的类别信息来推断未知的类别。[1]

另一种方法是利用属性(attribute)来表示类别。假设我们有一个属性矩阵 \mathbf{A} ，其中每一行代表一个类别的属性向量。例如，对于动物属性，我们可以定义一个 4 维向量 (animalness, abstractness, sourness, yellowness)。那么，bee 的属性向量可能是 [1, 0, 0, 1]，yellow 的属性向量可能是 [0, 1, 0, 1]，unicorn 的属性向量可能是 [1, 1, 0, 0]。通过计算这些属性向量之间的相似度，我们可以推断出最接近的类别。

第8章 支持向量机

支持向量机（Support Vector Machine, SVM）是一种强大的分类模型，广泛应用于文本分类、图像识别、生物信息学等领域。本章将详细介绍SVM的基本原理、数学模型及其在实际应用中的表现。

8.1 支持向量机的基本原理

支持向量机的核心思想是通过寻找一个最优的决策边界，将不同类别的数据点分隔开。这个决策边界被称为“最大间隔”（Maximum Margin）。在SVM中，我们通常使用“软间隔”（soft margin）来衡量模型的泛化能力，即允许数据点在一定程度上跨越决策边界，以换取更好的分类性能。

SVM的数学模型可以表示为一个二次规划问题。通过引入拉格朗日乘子，我们可以将这个问题转化为对偶问题，从而更容易地求解。图8.1(a)展示了SVM在二维空间中的决策边界和支撑向量（Support Vectors）。

图8.1(b)展示了SVM在非线性可分情况下的应用，通过核函数（Kernel Function）将数据映射到高维空间，使得数据在该空间中变得线性可分。这种技术被称为“核技巧”（Kernel Trick）。

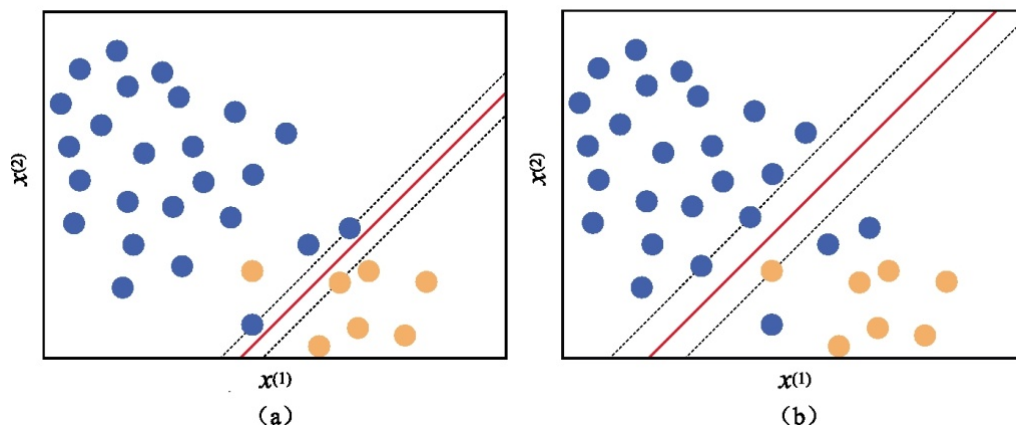


图8.1 数据分布图

(a)线性决策边界;(b)非线性决策边界

SVM模型在训练过程中，对于不平衡数据集，通常采用以下两种方法：

1. 过采样(oversampling)：通过复制或生成新的少数类样本，使数据集达到平衡。

2. 欠采样(undersampling)：通过删除部分多数类样本，使数据集达到平衡。

此外，还可以采用合成少数类过采样技术(synthetic minority oversampling technique, SMOTE)和自适应合成采样方法(adaptive synthetic sampling method, ADASYN)等方法。

SMOTE和ADASYN的原理如下：假设有一个少数类样本 \mathbf{x}_i ，其最近邻的 k 个多数类样本为 S_k ，则生成新的少数类样本 \mathbf{x}_{new} 的公式为：

$$\mathbf{x}_{\text{new}} = \mathbf{x}_i + \lambda(\mathbf{x}_{z_l} - \mathbf{x}_i)$$

其中 λ 是一个在 $[0, 1]$ 范围内随机选取的数。

SMOTE 与 ADASYN 都是过采样方法。SMOTE 通过生成新的样本 \mathbf{x}_i 来平衡数据集，而 ADASYN 则根据样本的难易程度来生成新的样本。具体来说，SMOTE 通过选择 S_k 中的样本 \mathbf{x}_k 和 \mathbf{x}_i ，生成新的样本 \mathbf{x}_{new} 。

在生成新的样本时，SMOTE 会随机选择 S_k 中的样本 \mathbf{x}_k 和 \mathbf{x}_i ，并生成新的样本 \mathbf{x}_{new} 。而 ADASYN 则会根据样本的难易程度来生成新的样本，使得模型能够更好地学习。

8.2 集成学习

集成学习是一种通过组合多个弱分类器来提高模型性能的方法。常见的集成学习方法包括 Bagging、Boosting 和 Stacking。Bagging 通过并行训练多个模型并取平均来降低方差；Boosting 通过串行训练多个模型并加权组合来提高模型的鲁棒性；Stacking 则通过训练多个模型并将它们的输出作为输入来训练一个元模型。

在 Bagging 中，我们通常会训练 3 个模型（averaging），然后通过 majority vote 来预测结果。而在 Stacking 中，我们会训练多个模型，并将它们的输出作为输入来训练一个元模型。

在 Boosting 中，我们会训练多个模型，并将它们的输出加权组合。具体来说，我们会训练一个 base model，然后根据其输出 \mathbf{x} 来训练下一个模型，直到达到预定的性能目标。

在 Stacking 中，我们会训练多个模型，并将它们的输出作为输入来训练一个元模型。具体来说，我们会训练 n 个模型 f_1, f_2, \dots, f_n ，然后将它们的输出 $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ 作为输入来训练元模型 g 。

在 Boosting 中，我们会训练多个模型 f_1, f_2, \dots, f_n ，并将它们的输出加权组合。具体来说，我们会训练一个模型 f_1 ，然后根据它的输出 \mathbf{x}_1 来训练下一个模型 f_2 ，直到达到预定的性能目标。

在 Stacking 中，我们会训练多个模型 f_1, f_2, \dots, f_n ，并将它们的输出作为输入来训练一个元模型 g 。具体来说，我们会训练 n 个模型 f_1, f_2, \dots, f_n ，然后将它们的输出 $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ 作为输入来训练元模型 g 。

[illegible][illegible][illegible]

8.4

`nn.Dropout(0.5)`, `nn.L1Loss`, `nn.BatchNorm1d`: `nn`

(dropout) `nn.Dropout(0.5)` (early stopping) `nn.L1Loss` (batch normalization) `nn.BatchNorm1d`, `nn.L1Loss`

[illegible]

0000000000000000,0000000000000000,0000000000000000
 4000000000000000,000000,0000000000000000000000000000
 0,00000 e 000000,0000000000:000000,0000000000000000000000

训练过程中,模型参数会不断更新,模型性能也会随之提高,但模型性能提高的同时,模型的体积也会随之增大,模型体积增大后,模型的推理速度也会随之降低,因此,在训练过程中,需要定期保存模型的checkpoint,以便在模型性能下降时,可以恢复到之前的状态;同时,也需要定期保存模型的参数,以便在模型性能下降时,可以恢复到之前的状态。

在训练过程中,模型的参数会不断更新,模型的体积也会随之增大,模型的推理速度也会随之降低,因此,在训练过程中,需要定期保存模型的checkpoint,以便在模型性能下降时,可以恢复到之前的状态;同时,也需要定期保存模型的参数,以便在模型性能下降时,可以恢复到之前的状态。

在训练过程中,模型的参数会不断更新,模型的体积也会随之增大,模型的推理速度也会随之降低,因此,在训练过程中,需要定期保存模型的checkpoint,以便在模型性能下降时,可以恢复到之前的状态;同时,也需要定期保存模型的参数,以便在模型性能下降时,可以恢复到之前的状态。

8.5 数据增强

数据增强,是指通过多种手段,对原始数据进行加工,生成新的数据,以增加训练数据量,提高模型的泛化能力,同时,也可以防止模型过拟合。

数据增强,是指通过多种手段,对原始数据进行加工,生成新的数据,以增加训练数据量,提高模型的泛化能力,同时,也可以防止模型过拟合。

数据增强,是指通过多种手段,对原始数据进行加工,生成新的数据,以增加训练数据量,提高模型的泛化能力,同时,也可以防止模型过拟合。

数据增强,是指通过多种手段,对原始数据进行加工,生成新的数据,以增加训练数据量,提高模型的泛化能力,同时,也可以防止模型过拟合。

**□;CNN□□□□□□,□RNN□□□□□□□□□□,□□□□□□□□□□□□,□□□□□□□
□□□□□□□,□softmax□sigmoid□□□□□□□□□□□□□□□□,□□□□□□□□□□
□□□□□□□**

8.6

1. 在 $(\text{root}, \text{tag})$ 处，如果 tag 是左孩子，则

[illegible]

输入向量, 权重矩阵, 偏置向量, 1
 输入向量, 权重矩阵, 偏置向量, 1
 , ReLU, 权重矩阵, 偏置向量, C_1
 输入向量, 权重矩阵, 偏置向量, softmax
 , C_2 (输出向量)

00000000000000000000,00000000000000000000
 0000000000000000,000000000000(0,1)0000 γ ,0000000000
 $\gamma C_i + (1-\gamma)C_2$ 00000000,000000000000 γ 000

8.7 四角

이것이 바로, 이진 탐색을 사용하는 이유입니다. 이진 탐색은 매우 빠릅니다.

이진 탐색은, 이진 탐색을 사용하는 이유입니다. 이진 탐색은 매우 빠릅니다. 이진 탐색은, 이진 탐색을 사용하는 이유입니다. 이진 탐색은 매우 빠릅니다. 이진 탐색은, 이진 탐색을 사용하는 이유입니다. 이진 탐색은 매우 빠릅니다.

8.8 이진 탐색

이진 탐색은, 이진 탐색을 사용하는 이유입니다. 이진 탐색은 매우 빠릅니다. 이진 탐색은, 이진 탐색을 사용하는 이유입니다. 이진 탐색은 매우 빠릅니다.

이진 탐색 (analysis of algorithm) 이진 탐색을 사용하는 이유입니다. 이진 탐색은 매우 빠릅니다. 이진 탐색은, 이진 탐색을 사용하는 이유입니다. 이진 탐색은 매우 빠릅니다. 이진 탐색은, 이진 탐색을 사용하는 이유입니다. 이진 탐색은 매우 빠릅니다.

이진, 이진 탐색 1 이진 탐색 M 이진 탐색 S 이진 탐색을 사용하는 이유입니다. 이진 탐색은 매우 빠릅니다. 이진 탐색은, 이진 탐색을 사용하는 이유입니다. 이진 탐색은 매우 빠릅니다.

```
1 def find_max_distance(S):
2     result = None
3     max_distance = 0
4     for x1 in S:
5         for x2 in S:
6             if abs(x1 - x2) >= max_distance:
7                 max_distance = abs(x1 - x2)
8                 result = (x1, x2)
9     return result
```

이진 탐색, 이진 탐색을 사용하는 이유입니다. 이진 탐색은 매우 빠릅니다. 이진 탐색은, 이진 탐색을 사용하는 이유입니다. 이진 탐색은 매우 빠릅니다. 이진 탐색은, 이진 탐색을 사용하는 이유입니다. 이진 탐색은 매우 빠릅니다.

$5N^2$ (1 comparison, 2 abs, 2 assignment) operations
operations, time complexity $O(N^2)$, constant 5, 0

operations, operations:

```
1 def find_max_distance(S):
2     result = None
3     min_x = float("inf")
4     max_x = float("-inf")
5     for x in S:
6         if x < min_x:
7             min_x = x
8         if x > max_x:
9             max_x = x
10    result = (max_x, min_x)
11    return result
```

operations, operations S operations, operations $O(N)$, operations
operations (more efficient)

operations (polynomial), operations, $O(N^2)$
 $O(N)$ operations, $M \cdot N^2$ operations $M \cdot 1$ operations 2 operations, operations, $O(N^2)$ operations operations, operations $O(\log N)$ operations

operations, operations, operations (avoid using loops
whenever possible), operations Python operations, operations wx ,
operations:

```
1 import numpy
2 wx = numpy.dot(w, x)
```

operations:

```
1 WX = 0
2 for i in range(N):
3     WX += w[i]*x[i]
```

0000,0000000000000000000000000000,00000000(set)000
 00(list)000000Python,0000000000x000S000,000S00000000
 0,00S000000000000

Python(dict)(hashmap)(key-value pair)

numpy, scipy, scikit-learn, pandas, matplotlib, seaborn, plotly, bokeh, folium, geopandas, shapely, pyproj, pygeos, rtree, pygeotools, pygeotools(method) C

`generator(generator)`

Python cProfile

[illegible]

- multiprocessing
- PyPy, Numba, Python

[1] :

第9章 统计推断

本章主要介绍统计推断的基本概念和方法，包括参数估计、假设检验、方差分析、回归分析等。本章是统计推断的基础，也是后续章节学习的前提。

9.1 参数估计

参数估计(density estimation)是指根据样本数据推断总体参数的过程。参数估计分为点估计和区间估计。点估计是指用样本均值、样本方差等统计量来估计总体均值、总体方差等参数。区间估计是指用样本均值、样本方差等统计量来估计总体参数的置信区间。参数估计的常用方法有矩估计法、极大似然估计法、贝叶斯估计法等。本章主要介绍矩估计法和极大似然估计法。参数估计的常用分布有正态分布、指数分布、泊松分布等。本章主要介绍正态分布、指数分布、泊松分布的参数估计。参数估计的常用检验有t检验、F检验、卡方检验等。本章主要介绍t检验、F检验、卡方检验。参数估计的常用软件有SPSS、SAS、R等。本章主要介绍SPSS、SAS、R的参数估计。

设 $\{x_i\}_{i=1}^N$ 为独立同分布的随机变量，其概率密度函数(pdf)为 f ， $i=1, \dots, N$ 。若 $x_i \in \mathbb{R}$ ，则 f 满足 $f(x) \geq 0$ 且 $\int_{-\infty}^{\infty} f(x) dx = 1$ 。定义 \hat{f}_b 为 f 的核密度估计(kde)：

$$\hat{f}_b(x) = \frac{1}{Nb} \sum_{i=1}^N k\left(\frac{x - x_i}{b}\right)$$

(9.1)

我们选择 b 使得 $k(z)$ 满足 $k(0) = 1$ 且 $k(z) \geq 0$ 对任意 z 成立:

$$k(z) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right)$$

我们定义 b 为使得 \hat{f}_b 的均方误差 (mean integrated squared error, MISE) 最小:

$$\text{MISE}(b) = \mathbb{E} \left[\int_{\mathbb{R}} (\hat{f}_b(x) - f(x))^2 dx \right] \quad (9.2)$$

在 9.2 节中, 我们将证明 \hat{f}_b 的 MISE 可以表示为 $\sum_{i=1}^N \frac{1}{N} \mathbb{E}[(\hat{f}_b(x_i) - f(x_i))^2]$ 的形式, 其中 $\{x_i\}_{i=1}^N$ 是独立同分布的随机变量, 其分布为 f 。

因此, 我们只需要最小化 $\mathbb{E}[(\hat{f}_b(x) - f(x))^2]$ 即可。注意到 $\hat{f}_b(x) = \frac{1}{N} \sum_{i=1}^N \hat{f}_b^{(1)}(x_i)$, 其中 $\hat{f}_b^{(1)}(x_i) = \int_{\mathbb{R}} k_b(x - x_i) f(x) dx$ 。因此, 我们只需要最小化 $\mathbb{E}[(\hat{f}_b^{(1)}(x_i) - f(x_i))^2]$ 即可。

在 9.2 节中, 我们将证明:

$$\mathbb{E} \left[\int_{\mathbb{R}} \hat{f}_b^2(x) dx \right] - 2 \mathbb{E} \left[\int_{\mathbb{R}} \hat{f}_b(x) f(x) dx \right] + \mathbb{E} \left[\int_{\mathbb{R}} f(x)^2 dx \right]$$

可以表示为 b 的函数, 且当 $b \rightarrow 0$ 时, 该函数趋于 $\int_{\mathbb{R}} f^2(x) dx$ 。

因此, 我们只需要最小化 $\mathbb{E}[(\hat{f}_b^{(1)}(x_i) - f(x_i))^2]$ 即可。注意到 $\hat{f}_b^{(1)}(x_i) = \int_{\mathbb{R}} k_b(x - x_i) f(x) dx$, 其中 $k_b(x) = \frac{1}{b} k\left(\frac{x}{b}\right)$ 。因此, 我们只需要最小化 $\mathbb{E}[(\hat{f}_b^{(1)}(x_i) - f(x_i))^2]$ 即可。

9.2 聚类

聚类(clustering)是指将数据集中的对象按照相似性分组的过程。相似性是指对象之间的相似程度，通常用距离来衡量。距离越小，相似性越高。聚类分析的目的是发现数据中的潜在结构，为数据挖掘提供基础。

聚类分析可以分为有监督和无监督两种。有监督聚类是指事先知道数据的类别，通过聚类来验证或发现新的类别。无监督聚类是指事先不知道数据的类别，通过聚类来发现数据的潜在结构。

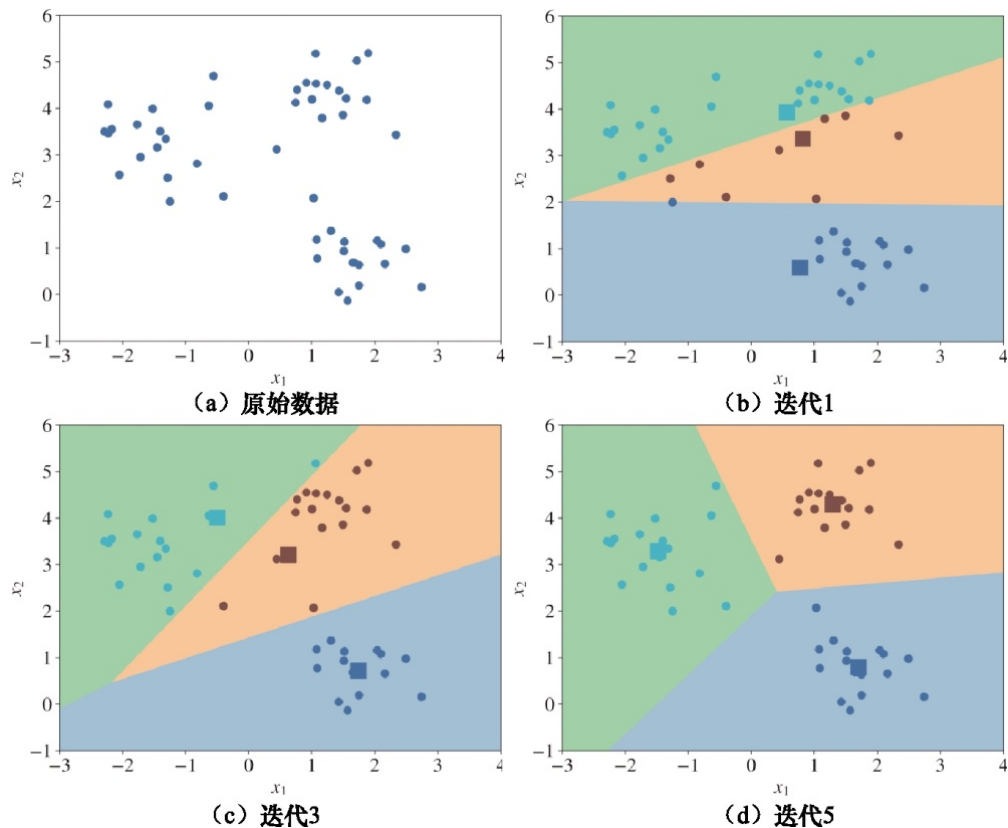
9.2.1 k-均值

k-均值(k-mean)是一种无监督聚类算法，它将数据集划分为k个簇(cluster)。每个簇有一个中心点(centroid)，称为簇心。算法通过迭代更新簇心和分配数据点到最近的簇心，直到收敛为止。k-均值算法的输入是数据集和簇数k，输出是k个簇的簇心和每个数据点的归属。

k-均值算法的步骤如下：1. 初始化k个簇心；2. 将每个数据点分配到最近的簇心；3. 计算每个簇心的新位置；4. 重复步骤2和3，直到簇心不再移动或移动距离小于某个阈值。

k-均值算法的优点是简单易懂，计算效率高。缺点是对于初始簇心的选择敏感，容易陷入局部最优解。可以通过多次运行或随机初始化来避免这个问题。

在9.2节中，我们将详细讨论k-均值算法的实现和性能。在9.2.1节中，我们将介绍k-均值算法的变种和实际应用。



9.2 $k=3$, k 的确定

对于 k 的确定问题, 有很多种方法来确定 k , 其中比较常用的方法有“肘部法则”, 即通过观察随着 k 的增加, 聚类误差 (如平方误差和 SSE) 的变化情况, 当误差下降到一个拐点后, 再增加 k , 误差下降的幅度会明显变小, 这个拐点所对应的 k 值即为最佳的聚类数。

9.2.2 DBSCAN 与 HDBSCAN

DBSCAN 是一种基于密度的聚类算法, 它不需要预先指定 k 值, 而是通过两个参数 ϵ 和 n 来控制聚类过程。其中 ϵ 表示邻域半径, n 表示邻域内所需的最小点数。对于任意一点 x , 如果其邻域内至少有 n 个点, 则 x 被认定为核心点 (core point)。如果某点位于核心点的邻域内, 则它属于该核心点所代表的簇。DBSCAN 的优点是能够发现任意形状的簇, 并且对噪声点具有较强的鲁棒性。

1. 数据集 D 中任意两个点 x, y 之间的距离 $d(x, y)$ 小于等于 ϵ ，则 x, y 属于同一个簇。
2. 数据集 D 中任意两个点 x, y 之间的距离 $d(x, y)$ 大于 ϵ ，则 x, y 属于不同的簇。
3. 数据集 D 中任意两个点 x, y 之间的距离 $d(x, y)$ 大于 ϵ ，则 x, y 属于不同的簇。

DBSCAN 算法中， ϵ 表示邻域半径， k 表示邻域内点的个数。DBSCAN 算法中， ϵ 表示邻域半径， k 表示邻域内点的个数。DBSCAN 算法中， ϵ 表示邻域半径， k 表示邻域内点的个数。

HDBSCAN 算法中， ϵ 表示邻域半径， k 表示邻域内点的个数。HDBSCAN 算法中， ϵ 表示邻域半径， k 表示邻域内点的个数。HDBSCAN 算法中， ϵ 表示邻域半径， k 表示邻域内点的个数。

HDBSCAN 算法中， n 表示数据集的大小， k 表示邻域内点的个数。HDBSCAN 算法中， n 表示数据集的大小， k 表示邻域内点的个数。HDBSCAN 算法中， n 表示数据集的大小， k 表示邻域内点的个数。

9.2.3 预测性能

预测性能，是指模型对未知数据的预测能力。预测性能的好坏，取决于模型的复杂度和训练数据的质量。预测性能的好坏，取决于模型的复杂度和训练数据的质量。预测性能的好坏，取决于模型的复杂度和训练数据的质量。

预测性能 (prediction strength) 是指模型对未知数据的预测能力。预测性能的好坏，取决于模型的复杂度和训练数据的质量。预测性能的好坏，取决于模型的复杂度和训练数据的质量。预测性能的好坏，取决于模型的复杂度和训练数据的质量。

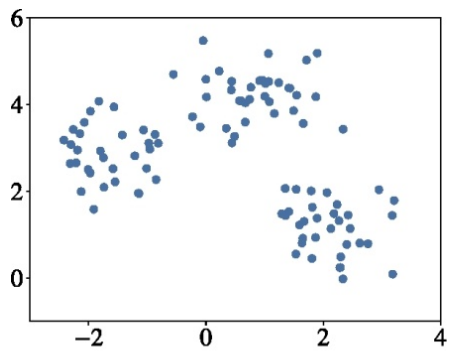
A 的复杂度为 $C(S_{tr}, k)$ 。 A 的复杂度为 $O(n^2)$ ，因此， A 的复杂度为 $O(n^2)$ 。

$N_{te} \times N_{te}$ (co-membership matrix) $\mathbf{D}[A, S_{te}]$: $\mathbf{x}_i, \mathbf{x}_{i'} \in A$, $\mathbf{D}[A, S_{te}]^{(i, i')} = 1$; $\mathbf{D}[A, S_{te}]^{(i, i')} = 0$

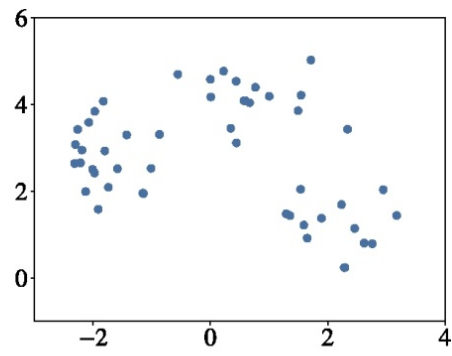
[illegible]

0000000000, 00 k 0000000000, 00000 $C(S_{te}, k)$ 0000000000
 0, 000 $C(S_{tr}, k)$ 0000000000000000, 00 k 000000(00000), 000000000
 000000000000000

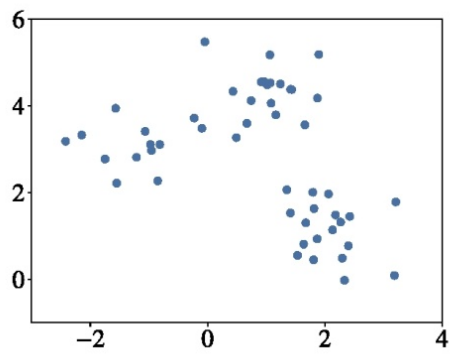
[illegible]



完整数据集

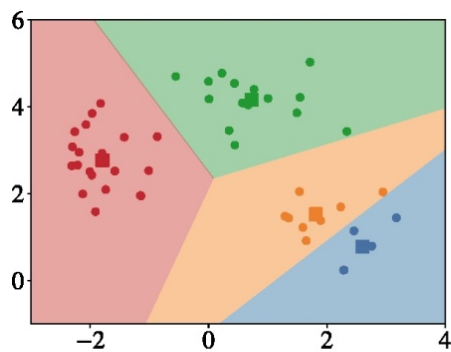


训练集

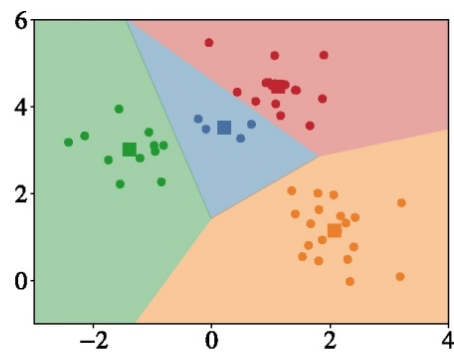


测试集

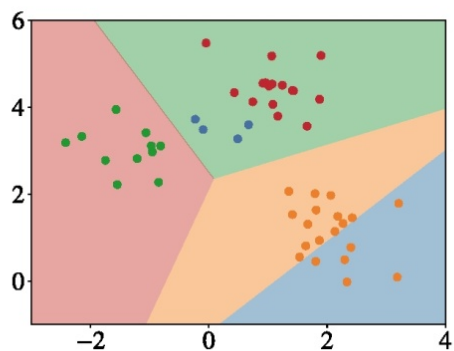
9.3 9.4



(a) 训练数据聚类方案



(b) 测试数据聚类方案



(c) 测试数据重叠于训练数据的聚类方案

9.4 $k=4$

给定 k , 定义:

$$ps(k) \stackrel{\text{def}}{=} \min_{j=1, \dots, k} \frac{1}{|A_j| (|A_j| - 1)} \sum_{i, i' \in A_j} D[A, S_{te}]^{(i, i')}$$

令 $A \stackrel{\text{def}}{=} C(S_{tr}, k)$ 为训练数据 $C(S_{tr}, k)$ 的聚类方案, $|A_j|$ 为 A_j 的簇大小

给定训练数据 $C(S_{tr}, k)$, 测试数据 $C(S_{te}, k)$, 定义:

测试数据重叠于训练数据的聚类方案

9.2.4 高维数据

DBSCAN 是一种基于密度的聚类算法 (hard clustering)，它不需要预先指定簇的数量。高维数据通常使用高斯混合模型 (Gaussian mixture model, GMM) 进行建模，高斯混合模型通过计算每个数据点的成员得分 (membership score) (HDBSCAN 使用) 来估计 GMM 参数。高斯混合模型 (GMM) 是一种概率模型，它假设数据是由多个高斯分布 (MND) 生成的，每个高斯分布代表一个簇。高斯混合模型的数学表达式如下：

$$f_X = \sum_{j=1}^k \phi_j f_{\mu_j, \Sigma_j}$$

其中， f_{μ_j, Σ_j} 表示高斯分布 (MND) j 的概率密度函数， ϕ_j 表示高斯分布 j 的权重， $j=1, \dots, k$ ， μ_j, Σ_j 表示高斯分布 j 的均值和协方差矩阵。高斯混合模型的参数估计通常使用期望最大化算法 (expectation maximization algorithm, EM) 进行，该算法通过最大化似然函数 (maximum likelihood) 来估计参数。

高斯混合模型的参数估计通常使用期望最大化算法 (EM) 进行，该算法通过最大化似然函数 (maximum likelihood) 来估计参数。高斯混合模型的参数估计通常使用期望最大化算法 (EM) 进行，该算法通过最大化似然函数 (maximum likelihood) 来估计参数。

$$f(x|\mu, \sigma_1^2) = \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp - \frac{(x - \mu_1)^2}{2\sigma_1^2},$$

$$f(x|\mu, \sigma_2^2) = \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp - \frac{(x - \mu_2)^2}{2\sigma_2^2}$$

(9.3)

高斯混合模型的参数估计通常使用期望最大化算法 (EM) 进行，该算法通过最大化似然函数 (maximum likelihood) 来估计参数。高斯混合模型的参数估计通常使用期望最大化算法 (EM) 进行，该算法通过最大化似然函数 (maximum likelihood) 来估计参数。

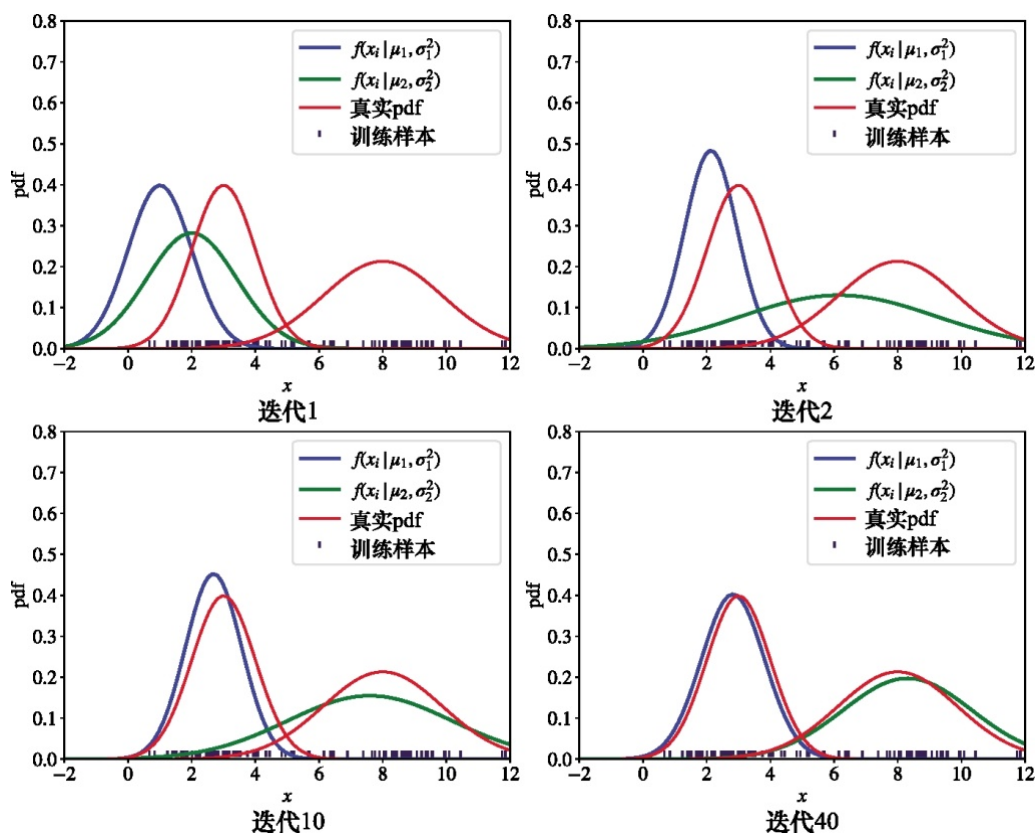
高斯混合模型的参数估计通常使用期望最大化算法 (EM) 进行，该算法通过最大化似然函数 (maximum likelihood) 来估计参数。高斯混合模型的参数估计通常使用期望最大化算法 (EM) 进行，该算法通过最大化似然函数 (maximum likelihood) 来估计参数。

(4) $\phi_j, j \in \{1, 2\}$

$$\phi_j \leftarrow \frac{1}{N} \sum_{i=1}^N b_i^{(j)}$$

初始化(1)(4), μ_j, σ_j^2 : 期望最大化: 参数估计 ϵ

9.6



9.6 EM $(k=2)$

EM k : 期望最大化, 参数估计 ϵ
 GMM x_i (soft clustering): x_i $b_i^{(j)}$
 9.4 μ_j, σ_j^2 (k), $b_i^{(j)}$

每个簇 j 的均值 μ_j 和方差 σ_j^2 ，每个数据点 x_i 属于簇 j 的概率 $p(x_i|\mu_j, \sigma_j^2)$ 。

当簇数 D 较大 ($D > 1$) 时，方差 σ^2 和协方差 Σ 的计算会变得复杂，此时可以使用最小距离法 (MND)。



k 均值聚类算法，GMM 模型参数估计通常使用期望最大化 (EM) 算法。

GMM 模型参数估计通常使用 k 均值聚类算法，每个数据点 x_i 属于簇 k 的概率 $f_{tr}^k(x_i)$ ，簇数 k 通常通过交叉验证确定。

$$\arg \max_k \prod_{i=1}^{|N_{te}|} f_{tr}^k(x_i)$$

其中 $|N_{te}|$ 是测试集的大小。

除了 GMM，还有其他聚类方法，如谱聚类 (spectral clustering)、层次聚类 (hierarchical clustering) 和 DBSCAN。HDBSCAN 是一种基于密度的聚类方法。

9.3 聚类

数据量较大,内存占用较高,因此需要采用分布式计算框架(如Hadoop、Spark)进行计算。同时,为了提高计算效率,可以采用GPU加速计算。在数据预处理阶段,需要计算每个数据点的特征向量,并将其存储在内存中。在训练阶段,需要计算每个数据点的损失函数,并将其存储在内存中。在推理阶段,需要计算每个数据点的预测结果,并将其存储在内存中。

在训练过程中,需要不断调整模型的参数,直到模型达到最优状态。在推理过程中,需要不断调整模型的参数,直到模型达到最优状态。在推理过程中,需要不断调整模型的参数,直到模型达到最优状态。

在训练过程中,需要不断调整模型的参数,直到模型达到最优状态。在推理过程中,需要不断调整模型的参数,直到模型达到最优状态。在推理过程中,需要不断调整模型的参数,直到模型达到最优状态。

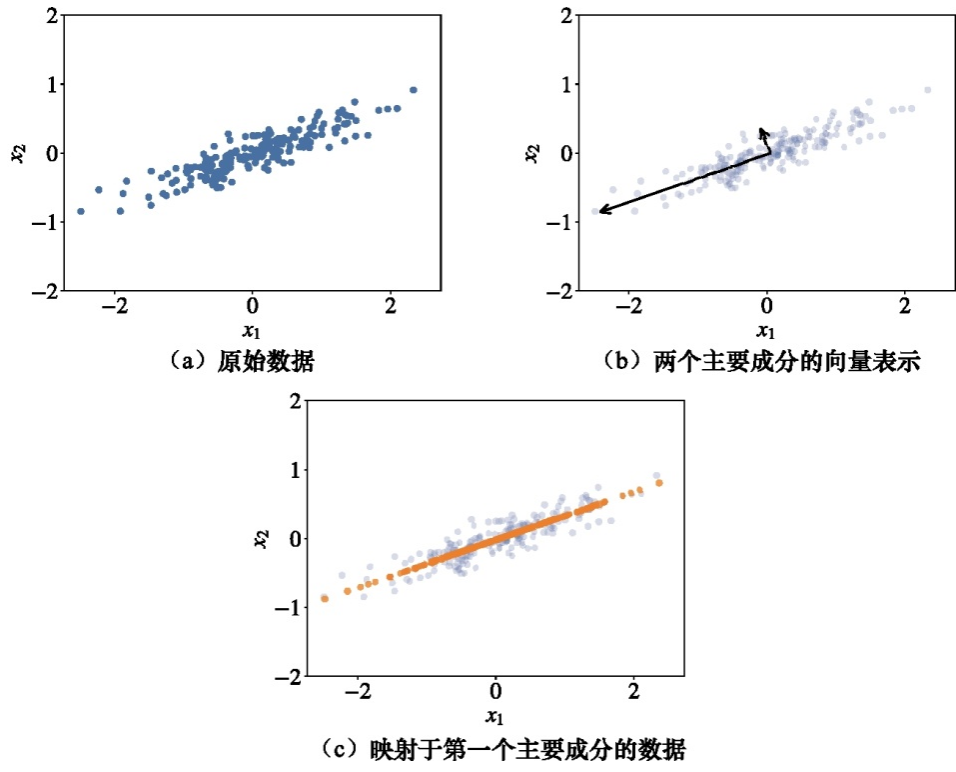
在训练过程中,需要不断调整模型的参数,直到模型达到最优状态。在推理过程中,需要不断调整模型的参数,直到模型达到最优状态。在推理过程中,需要不断调整模型的参数,直到模型达到最优状态。

9.3.1 数据预处理

在训练过程中,需要不断调整模型的参数,直到模型达到最优状态。在推理过程中,需要不断调整模型的参数,直到模型达到最优状态。在推理过程中,需要不断调整模型的参数,直到模型达到最优状态。

在训练过程中,需要不断调整模型的参数,直到模型达到最优状态。在推理过程中,需要不断调整模型的参数,直到模型达到最优状态。在推理过程中,需要不断调整模型的参数,直到模型达到最优状态。

3个主成分,前3个主成分,图9.7(b),主成分分析
 图,主成分分析图



9.7 PCA



图,主成分分析图 $D_{new} < D$, 图 D_{new} 主成分分析, 主成分分析
 主成分分析图, 图 $D_{new} = 1$, 主成分分析图
 9.7(c) 主成分分析

0000000000,0000000000,000000:0000000000000000000000
 000,0000000000,0002003000000000000000000000,002D03D00,
 00000000000000000000

9.3.2 UMAP

000000000000,0000000000000000000000,0**t-SNE**0**UMAP**,000
 000000000000000000000000000000000000,000000000000,00000000000
 000000000000,0000000000000000

0UMAP0,0000000 w 000:

$$w(\mathbf{x}_i, \mathbf{x}_j) \stackrel{\text{def}}{=} w_i(\mathbf{x}_i, \mathbf{x}_j) + w_j(\mathbf{x}_j, \mathbf{x}_i) - w_i(\mathbf{x}_i, \mathbf{x}_j) w_j(\mathbf{x}_j, \mathbf{x}_i) \quad (9.5)$$

00 $w_i(\mathbf{x}_i, \mathbf{x}_j)$ 000:

$$w_i(\mathbf{x}_i, \mathbf{x}_j) \stackrel{\text{def}}{=} \exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_j) - \rho_i}{\sigma_i}\right)$$

00, $d(\mathbf{x}_i, \mathbf{x}_j)$ 00000000000000, ρ_i 0 \mathbf{x}_i 000000000000,0 σ_i 0 \mathbf{x}_i 000 k 0000
 00(k 0000000)0

009.5000000000,0000000000001,0000000000
 0, $w(\mathbf{x}_i, \mathbf{x}_j) = w(\mathbf{x}_j, \mathbf{x}_i)$ 0

w 和 w' 都是标量; w' 也是标量, 9.5
 节

模糊集(fuzzy set)是定义在集合 S
 上的函数 $\mu_S(x) \in [0, 1]$, 其中 $x \in S$ (membership
 strength) 满足 $\mu_S(x) \geq 0$, 且 $x \in S$ 时, $\mu_S(x) \leq 1$, $x \in S$
 时 $\mu_S(x) = 1$, 且 S 是有限集

模糊集的交集

设 w, w' 都是 $[0, 1]$ 上的函数, 定义 $w(\mathbf{x}_i, \mathbf{x}_j)$ 和 $w'(\mathbf{x}_i, \mathbf{x}_j)$ 都是标量
 模糊集的交叉熵(fuzzy set
 cross-entropy), 为:

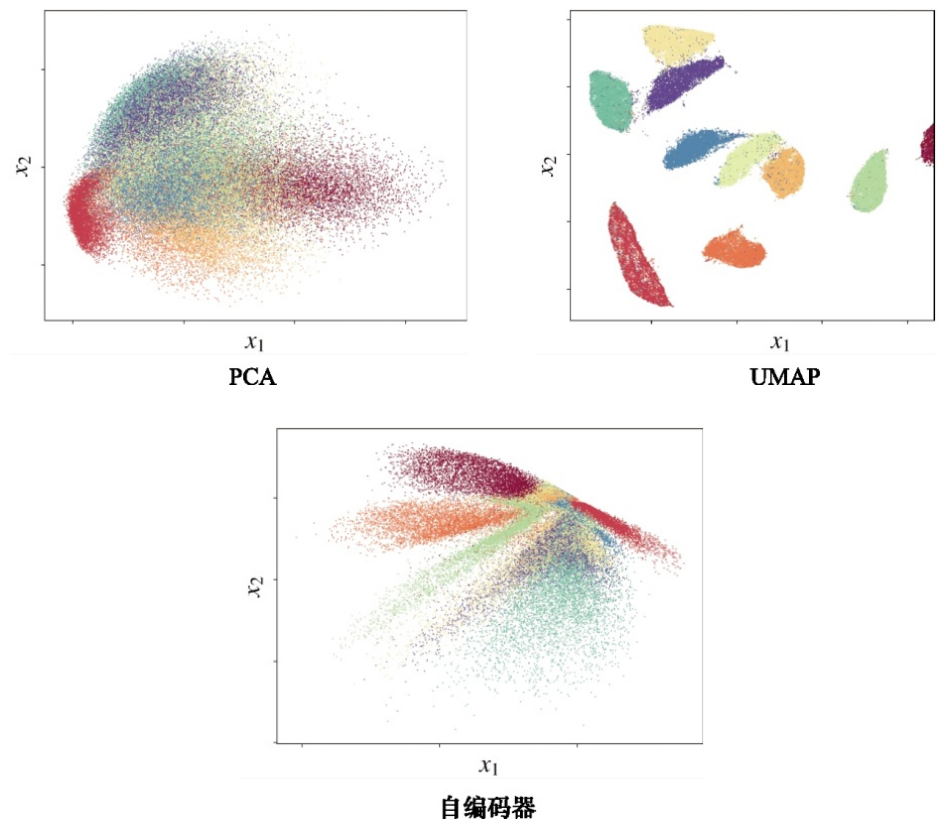
$$\begin{aligned}
 C_{w, w'} = \sum_{i=1}^N \sum_{j=1}^N \left[w(\mathbf{x}_i, \mathbf{x}_j) \ln \left(\frac{w(\mathbf{x}_i, \mathbf{x}_j)}{w'(\mathbf{x}_i, \mathbf{x}_j)} \right) + \right. \\
 \left. (1 - w(\mathbf{x}_i, \mathbf{x}_j)) \ln \left(\frac{1 - w(\mathbf{x}_i, \mathbf{x}_j)}{1 - w'(\mathbf{x}_i, \mathbf{x}_j)} \right) \right]
 \end{aligned}
 \tag{9.6}$$

设 \mathbf{x}' 是 \mathbf{x} 的“邻居”

9.6 节, 设 $\mathbf{x}'_i (i=1, \dots, N)$, 都是标量, 且 $C_{w, w'}$

9.8 节, 设 MNIST 数据集, MNIST 数据集
 大小为: 70 000 个样本, 9.8 节 10 个样本, 10 个样本

0000000000000000,UMAP0000000000(0000000000)0000
 0,UMAP00PCA00,0000000000



9.8 0MNIST000000300000000000

9.4 00000

00000(outlier detection)0000000000000000000000000000
 0000000000000000000000000000:00000000000000

000000000,0000000000000000000,0000000000000000
 0,000

00000000,0000000000000000000,00000000

第10章 相似性

10.1 距离

通常,我们使用欧氏距离(Euclidean distance)和余弦相似度(cosine similarity)来衡量两个向量的相似性,欧氏距离和余弦相似度都是标量,而欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量

欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量

欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量

$$d(x, x') = \|x - x'\| \stackrel{\text{def}}{=} \sqrt{(x - x')^2} = \sqrt{(x - x')(x - x')}$$

欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量

$$d_A(x, x') = \|x - x'\|_A \stackrel{\text{def}}{=} \sqrt{(x - x')^T A (x - x')}$$

欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量,欧氏距离和余弦相似度都是标量

$$A \stackrel{\text{def}}{=} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

矩阵, A 为相似度矩阵, c 为相似度阈值 (相似度)



相似度矩阵, 相似度矩阵, 相似度矩阵 A 为相似度矩阵

相似度矩阵, 相似度矩阵 (one-shot learning) 相似度矩阵 (Siamese network) 相似度矩阵 (triplet loss) 相似度矩阵: 相似度矩阵 S 相似度矩阵 相似度矩阵; 相似度 D 相似度矩阵

相似度矩阵, 相似度矩阵, 相似度矩阵, 相似度矩阵, 相似度矩阵

10.2 相似度

相似度 (learning to rank) 相似度矩阵, 相似度矩阵, 相似度矩阵, 相似度矩阵, 相似度矩阵, 相似度矩阵 M 相似度矩阵, 相似度矩阵 X_i 相似度矩阵 r_i 相似度矩阵 (相似度矩阵) 相似度矩阵, 相似度矩阵, 相似度矩阵, 相似度矩阵, 相似度矩阵, 相似度矩阵 f , 相似度矩阵, 相似度矩阵, 相似度矩阵, 相似度矩阵, 相似度矩阵 f 相似度矩阵

$i=1, \dots, N$, 相似度矩阵 X_i 相似度矩阵: $X_i = \{(\mathbf{x}_{i,j}, y_{i,j})\}_{j=1}^{r_i}$ 相似度矩阵 $\mathbf{x}_{i,j}$ 相似度矩阵 $j=1, \dots, r_i$ 相似度矩阵, $\mathbf{x}_{i,j}^{(1)}$ 相似度矩阵; $\mathbf{x}_{i,j}^{(2)}$ 相似度矩阵

$$\text{精确度} = \frac{|\{\text{相关文档}\} \cap \{\text{搜索返回文档}\}|}{|\{\text{搜索返回文档}\}|}$$

[illegible]

(Average Precision,AveP)□□□□□□:

$$\text{AveP}(q) = \frac{\sum_{k=1}^n (P(k) \cdot \text{rel}(k))}{|\{ \text{相关文档} \}|}$$

n , $P(k)$, k , $\text{rel}(k)$, k , 1 , 0 , Q , MAP

$$\text{MAP} = \frac{\sum_{q=1}^Q \text{AveP}(q)}{Q}$$

□□, □□□□□□ LambdaMART □□□□□□□□□□, □□□□□□□□□□
 $h(\mathbf{x})$ □□□, □□□□□□ $f(\mathbf{x}_j, \mathbf{x}_k)$ □□□□ \mathbf{x}_j □□□□ \mathbf{x}_k □□□□ (□□□□□□□□□□) □□□□□□
 □□□□□□□□ α □□□□□□□□ □□□

$$f(\mathbf{x}_i, \mathbf{x}_k) \stackrel{\text{def}}{=} \frac{1}{1 + \exp(h(\mathbf{x}_i) - h(\mathbf{x}_k))\alpha}$$

[illegible]

推荐系统的基本方法:基于内容(content-based filtering)和协同过滤(collaborative filtering)

在推荐系统中,用户和物品都可以表示为特征向量,用户特征向量记为 u ,物品特征向量记为 x ,用户和物品的交互结果记为 y ,推荐的目标是根据用户特征 u 和物品特征 x 来预测交互结果 y

基于内容的推荐方法,其核心思想是:将用户和物品表示为特征向量,通过计算用户特征向量和物品特征向量的相似度来推荐物品,相似度越高,推荐越准确

例如,假设用户特征向量为 $u = [1, 0, 1, 0, 1]$,物品特征向量为 $x = [1, 1, 0, 1, 0]$,那么它们的点积为 $u \cdot x = 1 \times 1 + 0 \times 1 + 1 \times 0 + 0 \times 1 + 1 \times 0 = 1$,这个值可以用来衡量用户对物品的兴趣程度

在基于内容的推荐中,通常会将用户特征向量和物品特征向量进行归一化,使得它们的模长为1,这样可以避免特征向量的量级对点积结果产生影响,归一化后的点积结果就是余弦相似度,范围在 $[-1, 1]$ 之间

基于协同过滤的推荐方法:通过分析用户和物品的交互历史来推荐物品

协同过滤可以分为基于用户的协同过滤和基于物品的协同过滤,基于用户的协同过滤是通过找到与目标用户相似的用户,推荐他们喜欢的物品,基于物品的协同过滤是通过找到与目标物品相似的物品,推荐它们

矩阵分解方法:因子化机器(Factorization Machines, FM)和去噪自编码器(Denoising Autoencoders, DAE)

10.3.1 数据表示

在推荐系统中，数据表示是一个非常重要的问题，它直接影响到推荐算法的性能。

图10.1展示了用户、电影和电影评价的数据表示。用户表示为一个向量 \mathbf{x} ，其中 $x_1, x_2, x_3, \dots, x_{99}, x_{100}$ 分别表示用户对不同电影的评价。电影表示为一个向量 \mathbf{y} ，其中 $y_1, y_2, y_3, \dots, y_{99}, y_{100}$ 分别表示电影的不同属性。

	用户				电影					电影评价										y
	Ed	Al	Zak	...	It	Up	Jaws	Her	...	It	Up	Jaws	Her				
	x_1	x_2	x_3	...	x_{21}	x_{22}	x_{23}	x_{24}	...	x_{40}	x_{41}	x_{42}	x_{43}	...	x_{99}	x_{100}				
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	0.2	0.8	0.4	0	...	0.3	0.8	1	$y^{(1)}$		
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0.2	0.8	0.4	0	...	0.3	0.8	3	$y^{(2)}$		
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0.2	0.8	0.4	0	...	0.3	0.8	2	$y^{(3)}$		
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.7	0.1	...	0.35	0.78	3	$y^{(4)}$		
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.7	0.1	...	0.35	0.78	1	$y^{(5)}$		
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	0.8	0	0	0.6	...	0.5	0.77	4	$y^{(6)}$		
\vdots	\vdots	\vdots	\vdots												\vdots	\vdots	\vdots	\vdots		
$\mathbf{x}^{(D)}$	0	0	0	...	0	0	1	0	...	0	0	1	0	...	0.95	0.85	5	$y^{(D)}$		

图10.1 用户、电影和电影评价的数据表示

在图10.1中，用户表示为一个向量 \mathbf{x} ，其中 $x_1, x_2, x_3, \dots, x_{99}, x_{100}$ 分别表示用户对不同电影的评价。电影表示为一个向量 \mathbf{y} ，其中 $y_1, y_2, y_3, \dots, y_{99}, y_{100}$ 分别表示电影的不同属性。

在图10.1中，用户表示为一个向量 \mathbf{x} ，其中 $x_1, x_2, x_3, \dots, x_{99}, x_{100}$ 分别表示用户对不同电影的评价。电影表示为一个向量 \mathbf{y} ，其中 $y_1, y_2, y_3, \dots, y_{99}, y_{100}$ 分别表示电影的不同属性。

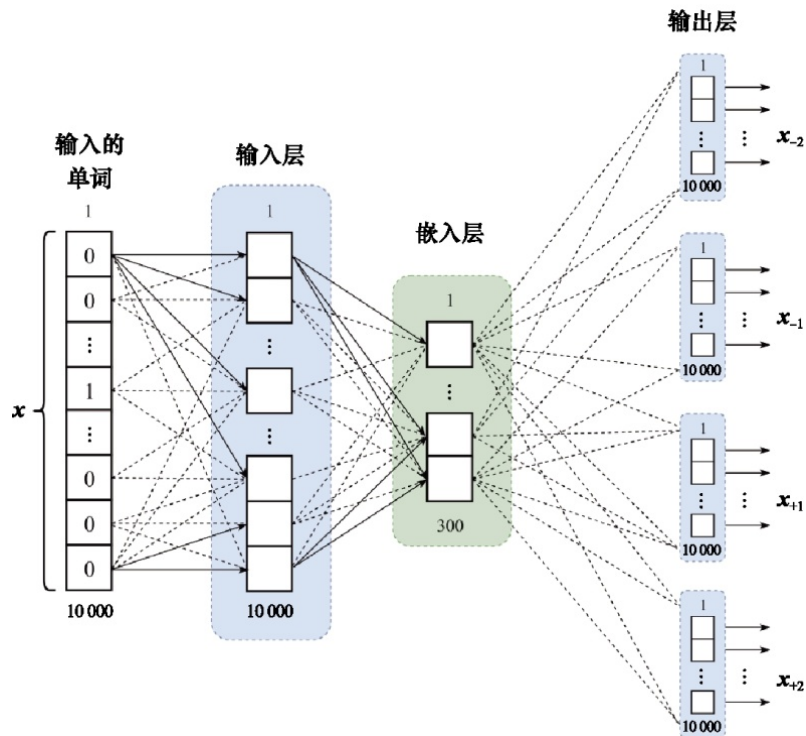


图10.2 词嵌入模型

词嵌入模型 (self-supervised): 词嵌入模型



词嵌入模型 softmax 词嵌入模型

word2vec[単語],埋め込みベクトル:ベクトル

```
softmax(hierarchicalsoftmax)(softmax_embeddings, softmax_embeddings)
(negativesampling)(softmax_embeddings, softmax_embeddings)
```

[1] $\sum_{i,j} D(D-1) w_{i,j}$

[2] $\angle \angle \square \square \square \square \square \square \square \square \square \square$

[3] α : $\alpha_1\alpha_2\alpha_3, \alpha_1\alpha_2\alpha_3\alpha_4\alpha_5, \alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_6\alpha_7\alpha_8, \alpha_1\alpha_2,$
 $\alpha_1\alpha_2\alpha_3\alpha_4, \alpha_1\alpha_2\alpha_3\alpha_4$

11 章

本章主要介绍概率图模型中的贝叶斯网络、马尔可夫随机场和条件随机场。

贝叶斯网络是一种有向无环图，用于表示变量之间的条件独立性。马尔可夫随机场是一种无向图，用于表示变量之间的相互依赖关系。条件随机场是一种有向图，用于表示变量之间的条件独立性，特别是在序列标注任务中。

本章将详细讨论这些模型的数学表示、推理算法和应用。

11.1 贝叶斯网络

贝叶斯网络是一种有向无环图，用于表示变量之间的条件独立性。它由节点和边组成，节点代表随机变量，边代表变量之间的直接依赖关系。贝叶斯网络的联合概率分布可以通过每个节点的局部概率分布和条件概率分布的乘积来计算。

11.2 马尔可夫随机场

马尔可夫随机场是一种无向图，用于表示变量之间的相互依赖关系。它由节点和边组成，节点代表随机变量，边代表变量之间的直接依赖关系。马尔可夫随机场的联合概率分布可以通过每个节点的局部概率分布和条件概率分布的乘积来计算。

11.3 广义线性模型

广义线性模型(Generalized Linear Model, GLM)是线性模型的扩展,它允许响应变量具有非正态分布,并且允许链接函数将线性预测与响应变量的期望值联系起来。GLM模型通常用于分类、计数和连续数据,是许多统计软件包中的标准模型。

11.4 图模型

图模型(Probabilistic Graphical Model, PGM)是一种用于表示概率分布的图论模型。它由节点和边组成,节点代表随机变量,边代表变量之间的条件独立性。图模型可以分为有向图模型(DAG)和无向图模型(UGM)。有向图模型通常用于表示因果关系,而无向图模型通常用于表示对称关系。图模型在机器学习中有着广泛的应用,特别是在自然语言处理和计算机视觉领域。

PGM模型可以分为生成模型和判别模型。生成模型学习数据的联合概率分布,而判别模型学习给定输入特征的条件概率分布。生成模型通常用于生成新数据,而判别模型通常用于分类和回归任务。图模型在机器学习中有着广泛的应用,特别是在自然语言处理和计算机视觉领域。图模型在机器学习中有着广泛的应用,特别是在自然语言处理和计算机视觉领域。图模型在机器学习中有着广泛的应用,特别是在自然语言处理和计算机视觉领域。

朴素贝叶斯 PGM, 贝叶斯网络 (Bayesian network) 信念网络 (belief network) 概率独立网络 (probabilistic independence network)

11.5

马尔可夫链蒙特卡罗(Markov Chain Monte Carlo, MCMC) 是一种用于从复杂的概率分布中采样的方法。它通过构建一个马尔可夫链，使得链的平稳分布与目标分布一致。MCMC 广泛应用于贝叶斯推断、统计物理、金融建模等领域。

11.6 1111

Genetic Algorithm (GA) is a search algorithm that mimics the process of natural selection. It is used to find the optimal solution to a problem by iteratively improving a population of candidate solutions. The process involves selection, crossover, and mutation. The fitness function is used to evaluate the quality of the solutions. The algorithm terminates when a satisfactory solution is found or a maximum number of iterations is reached.

GA 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 2086, 2087, 2088, 2089, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097, 2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2108, 2109, 2110, 2111, 2112, 2113, 2114, 2115, 2116, 2117, 2118, 2119, 2120, 2121, 2122, 2123, 2124, 2125, 2126, 2127, 2128, 2129, 2130, 2131, 2132, 2133, 2134, 2135, 2136, 2137, 2138, 2139, 2140, 2141, 2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149, 2150, 2151, 2152, 2153, 2154, 2155, 2156, 2157, 2158, 2159, 2160, 2161, 2162, 2163, 2164, 2165, 2166, 2167, 2168, 2169, 2170, 2171, 2172, 2173, 2174, 2175, 2176, 2177, 2178, 2179, 2180, 2181, 2182, 2183, 2184, 2185, 2186, 2187, 2188, 2189, 2190, 2191, 2192, 2193, 2194, 2195, 2196, 2197, 2198, 2199, 2200, 2201, 2202, 2203, 2204, 2205, 2206, 2207, 2208, 2209, 2210, 2211, 2212, 2213, 2214, 2215, 2216, 2217, 2218, 2219, 2220, 2221, 2222, 2223, 2224, 2225, 2226, 2227, 2228, 2229, 2230, 2231, 2232, 2233, 2234, 2235, 2236, 2237, 2238, 2239, 2240, 2241, 2242, 2243, 2244, 2245, 2246, 2247, 2248, 2249, 2250, 2251, 2252, 2253, 2254, 2255, 2256, 2257, 2258, 2259, 2260, 2261, 2262, 2263, 2264, 2265, 2266, 2267, 2268, 2269, 2270, 2271, 2272, 2273, 2274, 2275, 2276, 2277, 2278, 2279, 2280, 2281, 2282, 2283, 2284, 2285, 2286, 2287, 2288, 2289, 2290, 2291, 2292, 2293, 2294, 2295, 2296, 2297, 2298, 2299, 2300, 2301, 2302, 2303, 2304, 2305, 2306, 2307, 2308, 2309, 2310, 2311, 2312, 2313, 2314, 2315, 2316, 2317, 2318, 2319, 2320, 2321, 2322, 2323, 2324, 2325, 2326, 2327, 2328, 2329, 2330, 2331, 2332, 2333, 2334, 2335, 2336, 2337, 2338, 2339, 2340, 2341, 2342, 2343, 2344, 2345, 2346, 2347, 2348, 2349, 2350, 2351, 2352, 2353, 2354, 2355, 2356, 2357, 2358, 2359, 2360, 2361, 2362, 2363, 2364, 2365, 2366, 2367, 2368, 2369, 2370, 2371, 2372, 2373, 2374, 2375, 2376, 2377, 2378, 2379, 2380, 2381, 2382, 2383, 2384, 2385, 2386, 2387, 2388, 2389, 2390, 2391, 2392, 2393, 2394, 2395, 2396, 2397, 2398, 2399, 2400, 2401, 2402, 2403, 2404, 2405, 2406, 2407, 2408, 2409, 2410, 2411, 2412, 2413, 2414, 2415, 2416, 2417, 2418, 2419, 2420, 2421, 2422, 2423, 2424, 2425, 2426, 2427, 2428, 2429, 2430, 2431, 2432, 2433, 2434, 2435, 2436, 2437, 2438, 2439, 2440, 2441, 2442, 2443, 2444, 2445, 2446, 2447, 2448, 2449, 2450, 2451, 2452, 2453, 2454, 2455, 2456, 2457, 2458, 2459, 2460, 2461, 2462, 2463, 2464, 2465, 2466, 2467, 2468, 2469, 2470, 2471, 2472, 2473, 2474, 2475, 2476, 2477, 2478, 2479, 2480, 2481, 2482, 2483, 2484, 2485, 2486, 2487, 2488, 2489, 2490, 2491, 2492, 2493, 2494, 2495, 2496, 2497, 2498, 2499, 2500, 2501, 2502, 2503, 2504, 2505, 2506, 2507, 2508, 2509, 2510, 2511, 2512, 2513, 2514, 2515, 2516, 2517, 2518, 2519, 2520, 2521, 2522, 2523, 2524, 2525, 2526, 2527, 2528, 2529, 2530, 2531, 2532, 2533, 2534, 2535, 2536, 2537, 2538, 2539, 2540, 2541, 2542, 2543, 2544, 2545, 2546, 2547, 2548, 2549, 2550, 2551, 2552, 2553, 2554, 2555, 2556, 2557, 2558, 2559, 2560, 2561, 2562, 2563, 2564, 2565, 2566, 2567, 2568, 2569, 2570, 2571, 2572, 2573, 2574, 2575, 2576, 2577, 2578, 2579, 2580, 2581, 2582, 2583, 2584, 2585, 2586, 2587, 2588, 2589, 2590, 2591, 2592, 2593, 2594, 2595, 2596, 2597, 2598, 2599, 2600, 2601, 2602, 2603, 2604, 2605, 2606, 2607, 2608, 2609, 2610, 2611, 2612, 2613, 2614, 2615, 2616, 2617, 2618, 2619, 2620, 2621, 2622, 2623, 2624, 2625, 2626, 2627, 2628, 2629, 2630, 2631, 2632, 2633, 2634, 2635, 2636, 2637, 2638, 2639, 2640, 2641, 2642, 2643, 2644, 2645, 2646, 2647, 2648, 2649, 2650, 2651, 2652, 2653, 2654, 2655, 2656, 2657, 2658, 2659, 2660, 2661, 2662, 2663, 2664, 2665, 2666, 2667, 2668, 2669, 2670, 2671, 2672, 2673, 2674, 2675, 2676, 2677, 2678, 2679, 2680, 2681, 2682, 2683, 2684, 2685, 2686, 2687, 2688, 2689, 2690, 2691,

[illegible]

00000000000000000000,GA000000000000000000,00000
 0000000000000000

11.7 1111

强化学习(Reinforcement Learning, RL)是一种机器学习范式，旨在通过试错学习最优策略。它通常涉及一个智能体(agent)与环境交互，通过接收奖励(reward)来学习最优策略。RL广泛应用于游戏、机器人、推荐系统等领域。

□□□□□, □Q□(Q-learning), □□□□□□□□□□, □□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□(□□)□□□□□□□□□□

[illegible]

問

A

accuracy(精度) 分類器の予測結果が正解である割合、つまり正解率を指す。
例

activation function(活性化関数) ニューラルネットワークの各層の出力を、非線形に変換する関数。例えば、 $\sigma(x) = \frac{1}{1 + e^{-x}}$ などの関数がある。

active learning(アクティブ学習) 学習データが不足している場合に、モデルが自らデータを要求する学習方法。例えば、モデルが自信が低いデータを要求し、それを追加して学習を行う。

AdaBoost AdaBoostは、弱分類器を組み合わせて強分類器を構築するアルゴリズム。例えば、複数の弱分類器の出力を重み付けして、最終的な分類結果を決定する。

agglomerative clustering algorithm(凝集型クラスタリングアルゴリズム) “ボトムアップ”のクラスタリング方法。例えば、各データ点を最初の一つのクラスタとし、徐々にクラスタを合併していく。

area under the ROC curve(ROC曲線の面積) 二値分類モデルの性能を評価するための指標。ROC曲線(Receiver Operating Characteristic curve)の面積(面積)を指す。例えば、ROC曲線が対角線に近いほど性能が低い。

autoencoder(오토인코더) 입력데이터, 출력데이터를 모두 생성하는, 입력 데이터를 입력받아, 출력 데이터를 생성하는, 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조, 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조, 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조

B

backpropagation(역전파) 입력데이터, 출력데이터를 모두 생성하는, 입력 데이터를 입력받아, 출력 데이터를 생성하는, 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조, 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조, 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조

bag of words(단어) 입력데이터를 입력받아, 출력 데이터를 생성하는, 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조, 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조, 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조, 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조

bagging(단어) 입력데이터를 입력받아, 출력 데이터를 생성하는, 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조, 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조, 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조, 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조

base learner(기본학습기) 입력데이터를 입력받아, 출력 데이터를 생성하는, 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조, 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조, 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조, 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조

baseline(기준) 입력데이터를 입력받아, 출력 데이터를 생성하는(heuristic) 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조, 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조, 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조, 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조

batch(배치) 입력데이터(minibatch), 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조, 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조, 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조, 입력 데이터를 입력받아, 출력 데이터를 생성하는 구조

batch normalization(`batch_norm`) `batch_norm, batch_norm_stats =`
`batch_norm(x, offset=0, scale=1, mean=(mu: float), var=(sigma2: float),`
`momentum=0.9, epsilon=1e-5, regularization)` `batch_norm, batch_norm_stats =`
`batch_norm(x, offset=0, scale=1, mean=(mu: float), var=(sigma2: float),`

Bayes' Rule($P(A|B)$) $\frac{P(B|A) \cdot P(A)}{P(B)}$, 1800年代の統計学・確率論
の発展に貢献した。

Bayesian optimization(BO) 超パラメータ最適化

bias() □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
 □□□□□□(□□□)□

[illegible]

```
bigram(())  # 빈 리스트를 전달하면 빈 리스트를 반환
"cat"을 bigram으로 생성하면 ["ca", "at"]이 생성됨
3번 반복: "ca", "at", "ca", "at", "ca", "at"
"ca"와 "at"가 3번씩 생성됨
```

[illegible]

binary classification() □□□□□,□□□□□□□□□□□□□□
 □□,□□□□□□□□□□□□□□□□□□□□□□

binary variable(二値変数) 値が二つの変数(変数値):1と0,“真”と“偽”

binning(ビン) 変数値,連続変数値を区間(区間値),区間値をビン区間値,区間値を区間値,区間値を区間値:区間値0~5区間値,6~10区間値,11~15区間値,16~25区間値,区間値

boosting(ブースト) 区間値,区間値を区間値(区間値)区間値,区間値を区間値区間値,区間値を区間値区間値,区間値AdaBoost区間値

bootstrap aggregating(ブートストラップアグリゲーション) Bagging(バギング)

Bootstrapping(ブートストラップ) 区間値,区間値区間値,区間値区間値,区間値区間値,区間値区間値,区間値区間値,区間値区間値

Bottleneck(ボトルネック) 区間値,区間値区間値,区間値区間値,区間値区間値,区間値区間値,区間値区間値

bucket(バケット) bin(ビン)

bucketing(バケット) binning(ビンニング)

C

C4.5区間値区間値区間値ID3,区間値:区間値区間値区間値区間値区間値,区間値区間値“区間値”区間値区間値

CART(□□□□□) □□□□□□□□□□□□, □C4.5□□□

categorical feature(`cat`) `cat`

categorical variable(□□□□) □□□□□□□□□□,□□□□□□□□□□

□□,□□□□“□□□□□”□□□3□□□□:“□”“□”□“□”□

centroid(\mathbf{X}) k \mathbf{X} k $k=4, k=1$

□k□□□□□□4□□□□

```
class( )
```

00000000000000000000,00000000000000,0000000“0000”0“00
 000”0000000000000000,000000“0”“0”“00”00

```
classification( )  0000000000000000
```

```
classification_algorithm(□□□□) □□□□□□□□, □□□□□□□□□□
```

□ □ □ □ □ □ □ □ □ □

```
classification_model(□□□□) □classification(□□)□
```

classification threshold(□□□□) □□□□,□□□□□□□□□□□□

, :

[illegible]

```
classifier(model,classification(xx,yy))
```

Cluster(`[[[]]]`) `[[[]]]`(`[[[]]]`)

convolution filter(**kernel**) 2D array, 2D array of 2D arrays
2D array of 2D arrays

convolutional neural network(**CNN**, **kernel**) 2D array
2D array, 2D array of 2D arrays of 2D arrays, CNN 2D array of 2D arrays of 2D arrays
2D array, 2D array of 2D arrays of 2D arrays of 2D arrays, 2D array of 2D arrays of 2D arrays
2D array of 2D arrays of 2D arrays

cosine similarity(**vector**) 2D array of 2D arrays, 2D array of 2D arrays
2D array of 2D arrays

cost function(**model**) 2D array of 2D arrays, 2D array of 2D arrays, 2D array of 2D arrays
(2D array of 2D arrays) 2D array of 2D arrays

cross entropy(**model**) 2D array of 2D arrays of 2D arrays X 2D array of 2D arrays p
2D array of 2D arrays $H(p, q) = -\sum_{x \in X} p(x) \log q(x)$ 2D array of 2D arrays of 2D arrays, 2D array
2D array of 2D arrays

cross-validation(**model**) 2D array of 2D arrays of 2D arrays of 2D arrays
2D array of 2D arrays, 2D array of 2D arrays, 2D array of 2D arrays of 2D arrays of 2D arrays
2D array of 2D arrays of 2D arrays of 2D arrays

D

dataset(**model**) 2D array of 2D arrays

DBSCAN 2D array of 2D arrays, 2D array of 2D arrays of 2D arrays of 2D arrays

decision boundary(決定境界) 特徴空間を分割する境界線、超平面。異なるクラスに属するデータ点を分離するための境界。

decision tree(決定木) 特徴空間を分割する木構造のモデル。根ノードから葉ノードまで、分割条件(特徴と閾値)に基づいてデータを分割。代表的なアルゴリズムとしてID3、C4.5、CARTがある。

decision tree learning algorithm(決定木学習アルゴリズム) 決定木を構築するためのアルゴリズム。ID3、CART、C4.5などが代表的。

deep neural network(深層ニューラルネットワーク) 複数の隠れ層を持つニューラルネットワーク。画像認識、音声認識などに広く利用される。

density-based clustering algorithm(密度ベースのクラスタリングアルゴリズム) データの密度に基づいてクラスタを形成するアルゴリズム。DBSCANが代表的。

dimensionality reduction(次元削減) 高次元データを低次元に変換する技術。特徴抽出、可視化などに利用される。代表的な手法としてPCA(Principal Component Analysis)、t-SNE、UMAPなどがある。word2vecは単語をベクトルで表現する技術で、次元削減の一種と見られる。

divisive clustering algorithm(分割クラスタリングアルゴリズム) データを分割してクラスタを形成するアルゴリズム。例: 階層的クラスタリング、DBSCAN。

dropout() 在神经网络中，随机地使一部分神经元失效，即设置为0，以防止模型过拟合。

dummy variable(虚拟变量) 在回归分析中，用于表示定性变量的变量。通常取值为0或1，0表示不属于某类，1表示属于某类。例如，在binning(分箱)中，0表示不属于某箱，1表示属于某箱。

E

early stopping(提前停止) 在训练过程中，当模型的性能不再提升时，提前停止训练，以防止模型过拟合。

embedding(嵌入) 将离散数据（如单词）转换为连续向量表示。例如，word2vec、GloVe等模型，可以将单词转换为高维向量，使得语义相似的单词在向量空间中距离较近。

ensemble learning(集成学习) 通过组合多个模型的预测结果来提高模型的泛化能力。

ensemble learning algorithm(集成学习算法) 用于实现集成学习的算法，如Bagging、Boosting、Random Forest等。

epoch(epoch) 在训练过程中，一个完整的数据集遍历称为一个epoch。

evaluation metric(评价指标) 用于衡量模型性能的标准，如ROC、F1 score等。

evolutionary algorithm(进化算法) 模拟自然选择过程的优化算法，如遗传算法、粒子群优化等。

false positive(FP,FP) 分類器が正例と判断したもののうち、実際は負例であるもの、誤検出率(誤検出率)、誤検出率

feature(特徴) 入力変数、説明変数、予測変数、目的変数、ターゲット変数、アウトカム変数、アウトカム変数: 年齢(年齢)、性別(性別)、収入(収入)等

feature engineering(特徴工学) 特徴(特徴)を抽出・変換・選択・削除・生成する工程、特徴工学

feature selection(特徴選択) 特徴(特徴)を選択・削除する工程

feature vector(特徴ベクトル) 特徴(特徴)の集合

feature,informative(特徴,情報性) 特徴(特徴)の情報性

feedforward neural network(FFNN,前向きネットワーク) 入力層、隠れ層、出力層の3層構造を持つニューラルネットワーク、前向きネットワーク、前向きネットワーク(前向きネットワーク)、前向きネットワーク

few-shot learning(少ショット学習) 学習データが非常に少ない状況での学習、少ショット学習

G

gate(ゲート) 入力変数、説明変数、予測変数、目的変数、ターゲット変数、アウトカム変数

gated recurrent unit(GRU,ゲートリカレントユニット) 入力変数、説明変数、予測変数、目的変数、ターゲット変数、アウトカム変数、GRUはリカレントニューラルネットワークの一種、ゲートリカレントユニット

GRU() () ()

gated unit() , “ ” (GRU)(LSTM)

Gaussian distribution() ,
$$f_{\mu, \sigma^2}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \mu, \sigma^2$$

Gaussian process() ,

generalization() ,

genetic algorithm(GA,) , () , GA , “ ” “ ” “ ” d

GloVe() , word2vec

gradient boosting() ,

gradient clipping(梯度裁剪) 限制梯度的范数,防止梯度爆炸,通常使用L2范数进行裁剪,防止梯度过大导致模型训练不稳定

gradient descent(梯度下降) 一种优化算法,通过迭代更新模型参数,使得损失函数最小化,通常使用随机梯度下降法(Stochastic Gradient Descent, SGD)

gradient step(梯度步) 在梯度下降法中,每次迭代更新模型参数的步长,通常使用学习率来控制步长

grid search(网格搜索) 一种超参数优化方法,通过遍历所有可能的超参数组合,找到最优的模型性能

H

hidden layer(隐藏层) 神经网络中,位于输入层和输出层之间的层,用于提取特征

hidden unit(隐藏单元) 神经网络中,位于隐藏层中的单个神经元

hierarchical clustering algorithm(层次聚类算法) 一种聚类算法,通过递归地将数据点分组,形成树状结构,通常用于处理非凸数据集

hinge loss(合页损失) 一种损失函数,用于二分类问题,通过最大化正确分类的样本与错误分类的样本之间的间隔,通常用于支持向量机(SVM)

holdout set(测试集) 在机器学习中,用于评估模型性能的数据集,通常从训练集中划分出来,用于验证模型的泛化能力,防止过拟合

hyperparameter(超パラメータ) パラメータのうち、学習アルゴリズムの性能を大きく左右するものを指す。例えば、学習率、隠れ層のニューロン数、ドロップアウト率などが挙げられる。最適化: 学習率を調整することで、学習の収束を早めたり遅めたりできる。

hyperparameter tuning(超パラメータチューニング) 最適な超パラメータの組み合わせを見つけるプロセス。グリッドサーチ、ランダムサーチ、ベイズ最適化などが行われる。

Hyperplane(超平面) n次元空間で、n-1次元の平面を指す。例えば、2次元空間では直線、3次元空間では平面。サポートベクターマシン(SVM)では、異なるクラスを分離するための境界線として利用される。

I

ID3 決定木アルゴリズムの一種。情報増益を用いて最適な分割属性を選択する。

input layer(入力層) ニューラルネットワークの最初の層。入力データを処理する。

instance(インスタンス) 学習データの1つの例。特徴量とラベルで構成される。

instance-based learning(インスタンスベース学習) 学習データを個々のインスタンスとして扱う学習方法。k近傍法(k-NN)が代表的なアルゴリズム。

Iteration(反復) 学習プロセスを繰り返す1回のサイクル。例えば、バッチ学習では、バッチデータを1回処理する。

K

k Nearest Neighbor(kNN)(k-近隣) 入力データと、学習データ
の類似度を計算し、学習データの k 個の最も類似度の高いデータを
見つけ、その k 個のデータの平均値を出力する

k-mean(k-平均) 入力データを k 個のクラスに分割し、
各クラスの平均値を計算する

k-median(k-中央値) 入力データを k 個のクラスに分割し、
各クラスの中央値を計算する

Kernel() 入力データ x と学習データ x_j の類似度を計算する
関数

kernel trick() 入力データ x と学習データ x' の類似度を
計算する関数 $\phi(x)$ と $\phi(x')$ の内積を計算する

L

L1 regularization(L1正則化) 入力データを $L1$ ノルムで
正規化する

L2 regularization(L2正則化) 入力データを $L2$ ノルムで
正規化する

label() 入力データをラベル付けする関数

labeled example() 入力データをラベル付けする関数

Latent Dirichlet Allocation(LDA,)

latent semantic Indexing(라센스)

layer() ,

```
learning algorithm(inputs)    outputs,losses,weights
```

```
learning_rate(0.001)  # 学习率, 0.001
optimizer              # 优化器
```

linear regression(`data`) `data`, `data`

log loss(□□□□) □□□□□□□□□□□□□□□□

logistic regression(`logit`) `logit`,`logit`,`logit`
`logit`,`logit`,`logit`
`logit`,`logit`,`logit`

“ ” , “ ”

long short-term memory(LSTM)unit()

,LSTM LSTM

3 (“ ” “ ” “ ”)RNN ,

LSTM LSTM ,RNN

RNN

loss() 损失函数

loss function() 损失函数,用于衡量模型预测结果与真实结果之间的差异。通常用于监督学习任务中,如分类和回归任务。

LSTM network() 长短期记忆网络(LSTM)模型

M

machine learning() 机器学习,一种通过数据训练模型来自动学习和改进性能的技术。通常用于分类、回归、聚类、降维等任务。

machine learning algorithm() 机器学习算法,用于实现机器学习任务的算法。通常包括线性回归、逻辑回归、支持向量机、决策树、神经网络等。

maximum margin classification() 最大间隔分类,一种用于二分类任务的线性模型。通过最大化决策边界与最近数据点的距离来实现分类。

metric learning() 度量学习,一种通过定义合适的距离度量来学习模型参数的技术。通常用于图像识别、文本分类等任务。

minibatch() 小批量,用于训练模型的批次大小。

minibatch stochastic gradient descent() 小批量随机梯度下降,一种用于训练模型的优化算法。通过随机选择小批量数据来计算梯度,从而实现模型的参数更新。

model() 模型名称,模型参数列表,模型训练数据,模型训练方法,模型训练结果,模型训练时间,模型训练日志,模型训练环境,模型训练平台

model,classification() 模型名称,模型参数列表,模型训练数据,模型训练方法,模型训练结果,模型训练时间,模型训练日志,模型训练环境,模型训练平台

model,regression() 模型名称,模型参数列表,模型训练数据,模型训练方法,模型训练结果,模型训练时间,模型训练日志,模型训练环境,模型训练平台

multi-class classification() 模型名称,模型参数列表,模型训练数据,模型训练方法,模型训练结果,模型训练时间,模型训练日志,模型训练环境,模型训练平台

multi-class logistic regression() 模型名称,模型参数列表,模型训练数据,模型训练方法,模型训练结果,模型训练时间,模型训练日志,模型训练环境,模型训练平台

N

neural network() 模型名称,模型参数列表,模型训练数据,模型训练方法,模型训练结果,模型训练时间,模型训练日志,模型训练环境,模型训练平台

neuron() unit()

normalization() 模型名称,模型参数列表,模型训练数据,模型训练方法,模型训练结果,模型训练时间,模型训练日志,模型训练环境,模型训练平台

O

objective function(θ) θ is a vector of parameters, θ is a vector of parameters

[illegible]

outlier() □□□□□□□□□□□□□□□□

output layer(1000) 00000000,00000000


output unit(□□□) □□□□□□□□□□□□□□□□


overfitting(과적합) 학습 데이터에 지나치게 맞춰져, 새로운 데이터에 대한 일반화 능력이 떨어지는 현상

P

parameter() 00000000,0000000000000000,000000000000000000000000,000000000000000000000000

partitional clustering algorithm($\{x_1, \dots, x_n\}$) $\{C_1, \dots, C_k\}$
 $\{x_1, \dots, x_n\}$, $\{C_1, \dots, C_k\}$ k \leq k \leq k

pooling() 



population(人口) 人口数,人口数(人口)数,人口数(人口)数
人口数(人口)数,人口数(人口)数

precision(精度) 分類器の予測結果が正解である割合、正解と予測された正解の割合
計算式: $\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$ confusion matrix(混同行列)から計算可能

predictive model(予測モデル) 入力データから出力を予測するためのモデル

predictive power(予測力) モデルが未知のデータに対してどれだけ正確に予測できるかを表す指標
計算式: $\text{predictive power} = \frac{\text{predicted value} - \text{actual value}}{\text{predicted value} + \text{actual value}}$

principal component analysis(PCA, 主成分分析) 高次元データを低次元に変換するための手法
データの分散を最大化する方向にデータを投影し、PCAの主成分を抽出する。主成分はデータの主要な変異を説明する方向を示す。主成分分析は、データの次元削減、データの可視化、データのクラスタリングなどに利用される。

R

random forests(ランダムフォレスト) 複数の決定木を組み合わせて予測を行う手法
ランダムに選択された特徴量とランダムに選択されたデータサブセットを用いて、複数の決定木を構築し、それらの予測結果を平均して最終的な予測を行う。

random search(ランダムサーチ) 特徴量の組み合わせをランダムに選択してモデルの性能を評価する手法
ランダムに選択された特徴量の組み合わせに対して、モデルの性能を評価し、最も良い性能を示す組み合わせを選択する。

random variable(ランダム変数) 確率変数、ランダムな値を持つ変数
確率変数、ランダムな値を持つ変数、ランダムな値を持つ変数、ランダムな値を持つ変数

recall(再現率) 分類器が正解と予測した正解の割合、正解と予測された正解の割合
計算式: $\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$ confusion matrix(混同行列)から計算可能

recurrent neural network(RNN, 再帰型ニューラルネットワーク) 時系列データの予測や分類に利用されるニューラルネットワーク
時系列データの予測や分類に利用されるニューラルネットワーク、時系列データの予測や分類に利用されるニューラルネットワーク

regression(θ) 回归函数, 预测函数(θ)函数

regression algorithm(θ) 回归算法

regression model(θ) θ model, regression(θ , x)

regularization(λ) 正则化, 防止过拟合
L1正则化 L2正则化

reinforcement learning(θ) 强化学习, 一种机器学习
算法, 通过试错来学习, 通过奖励和惩罚来学习, 通过
策略来学习

representation learning(θ) 表示学习, 一种机器学习
算法

risk(θ) 风险

S

semi-supervised learning(θ) 半监督学习, 一种机器学习
算法

sigmoid function(S) sigmoid 函数, $(0, 1)$ 之间的
函数

similarity metric(θ) 相似性度量, 一种度量相似性的
方法, 如欧氏距离, 余弦距离

standardization(標準化) 平均値を0,標準偏差を1にする操作
 $\mu=0$
 $\sigma=1$ にする操作, μ (平均値), σ (標準偏差)

statistic(統計) 統計学,統計学

statistical model(統計モデル) model(モデル)

stochastic gradient descent(SGD,確率勾配降下法) 確率勾配降下法,確率勾配降下法,確率勾配降下法,SGD

strong classifier(強力な分類器) 強力な分類器,強力な分類器
強力な分類器

supervised learning(教師学習) 教師学習,教師学習
教師学習

Support Vector Machine(SVM,サポートベクターマシン) サポートベクターマシン,サポートベクターマシン
サポートベクターマシンSVM

T

target(ターゲット) ターゲット,ターゲット

test set(テストセット) テストセット:training set(トレーニングセット)
validation set(バリデーションセット)

token(トークン) トークン,トークン

tokenization(tokens) \rightarrow tokens 를 tokens_list 로 변환하는 과정
 tokens_list 은 tokens 의 tokens 를 tokens_list 로 변환한 것

topic modelling(tokens_list) \rightarrow tokens_list 에서 topic 를 추출하는 과정
 topic 은 tokens_list 에서 topic 를 추출한 것
(LDA) \rightarrow topic 을 추출하는 과정 (LSI) \rightarrow

training(tokens_list) \rightarrow tokens_list 을 tokens_list 로 변환하는 과정

training data(tokens_list) \rightarrow tokens_list set(tokens_list)

training example(tokens_list) \rightarrow example , $\text{training}(\text{tokens_list}, \text{tokens_list})$

training loss(tokens_list) \rightarrow tokens_list 에서 tokens_list 를 추출하는 과정

training set(tokens_list) \rightarrow tokens_list , tokens_list 을 tokens_list 로 변환하는 과정
 tokens_list : $\text{test set}(\text{tokens_list})$ \rightarrow $\text{validation set}(\text{tokens_list})$

transfer learning(tokens_list) \rightarrow tokens_list 에서 tokens_list 를 추출하는 과정
 tokens_list 은 tokens_list 의 tokens_list 를 추출한 것

true negative(TN, tokens_list) \rightarrow tokens_list 에서 tokens_list 를 추출하는 과정

true positive(TP, tokens_list) \rightarrow tokens_list 에서 tokens_list 를 추출하는 과정

U

UMAP(`data`) `data`,`data`

underfitting(`underfitting`) `underfitting`, `underfitting`

unit(□) □□□□□□□□,□□□□□□□□,□□□□□□□□,□□□□□□□□
□□□□□□□□□□□□□□(□□□□)□□□□□□□□□□□□□□

```
unlabeled example(□□□□) □example,unlabeled(□□,□
□□)□
```

unsupervised learning(教師なし学習) 教師なし学習(教師データなし)で学習する学習方法。教師データなしで学習する学習方法。

unsupervised learning algorithm()

V

validation example(□□□□) □□□□□□□□

[illegible]

validation set()

vanishing gradient problem(消失的梯度) 随着网络层数的增加，反向传播时，越靠近输入层的神经元，其梯度值越小，导致模型难以学习到输入与输出之间的有效关系。

通常用“ σ^2 ”,表示方差,表示数据分布的离散程度

variance(方差) 表示数据分布的离散程度,方差越大,数据分布越分散,方差越小,数据分布越集中(平均值)

volume(量) 表示数据量,表示数据分布的离散程度,通常,表示数据分布的离散程度(CNN)表示数据量

W

weak classifier(弱分类器) 表示数据,表示数据分布的离散程度,通常,表示数据分布的离散程度

word embedding(词嵌入) 表示数据分布的离散程度;通常,表示数据分布的离散程度,表示数据分布的离散程度

word2vec 表示数据分布的离散程度,表示数据分布的离散程度,表示数据分布的离散程度,表示数据分布的离散程度:GloVe(词嵌入)

Z

zero shot learning(零样本学习) 表示数据,表示数据分布的离散程度,表示数据分布的离散程度,表示数据分布的离散程度